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

КВАЛІФІКАЦІЙНА РОБОТА

Метод та система моніторингу параметрів комп'ютерної мережі в реальному часі на основі одноплатного комп'ютера
Назва теми

Рівень вищої освіти *другий (магістерський)*

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

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

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

Шифр КвРКІ 240120.24.01.19 ПЗ

Виконав здобувач II курсу, група КІ2м-24-1 — Підпис — Андрій МАРЦЕНЮК (Ініціали, прізвище)

Керівник — канд.-техн. наук, доцент (Науковий ступінь, учене звання) — Підпис — Катерина БЕРЕЗЬКА (Ініціали, прізвище)

Нормоконтролер — д. техн. наук, професор (Науковий ступінь, учене звання) — Підпис — Сергій ЛИСЕНКО (Ініціали, прізвище)

До захисту допускаю:
завідувач кафедри КІІС
«01» травня 2026 р.
дата — Підпис — Ольга ПАВЛОВА (Ініціали, прізвище)

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
Ольга ПАВЛОВА

" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Марценюку Андрію Володимировичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод та система моніторингу параметрів комп'ютерної мережі в реальному часі на основі одноплатного комп'ютера

Керівник проекту (роботи) Березька Катерина Миколаївна., канд.-техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6

2. Термін подання здобувачем роботи на кафедру 01.05.2026 р.

3. Вихідні дані до роботи Розробити метод та систему моніторингу параметрів мережі в реальному часі на основі одноплатного комп'ютера

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Теоретичні основи досліджуваної проблеми

Модель систем моніторингу параметрів комп'ютерної мережі

Метод і алгоритм системи моніторингу параметрів комп'ютерної мережі в реальному часі

Результати роботи методу і реалізації системи моніторингу параметрів комп'ютерної мережі в реальному часі. Експерименти здійснені відносно системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання « 12 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проєкту (роботи)	Термін виконання етапів проєкту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	12.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	15.01.2026	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою, постановка задачі	01.02.2026	виконано
4	Робота над розділом 2 – розробка моделі для вирішення поставленої задачі	01.03.2026	виконано
5	Робота над розділом 3 – розробка методу для вирішення поставленої задачі	29.03.2026	виконано
6	Робота над розділом 4 – проєктування та розробка реалізації, проведення експериментів, фіксація результатів	15.04.2026	виконано
7	Оформлення пояснювальної записки згідно вимог	25.04.2026	виконано
8	Попередній захист ВКР	26.04.2026	виконано
9	Захист ВКР на засіданні ЕК	травень 2026 року	

Здобувач _____ Андрій МАРІЄНЮК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Катерина БЕРЕЗЬКА
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Метод та система моніторингу параметрів комп’ютерної мережі в реальному часі на основі одноплатного комп'ютера

Автор роботи: Марценюк Андрій Володимирович

Керівник роботи: Березька Катерина Миколаївна

Пояснювальна записка: 81 с., 23 рис., 3 табл., 2 дод., 75 джерел.

МЕРЕЖІ, ПАРАМЕТРИ, ШВИДКОДІЯ, ОДНОПЛАТНІ КОМП’ЮТЕРИ, РЕСУРСИ, ДІАГНОСТИКА, АНАЛІЗ ЯКОСТІ.

Об’єктом дослідження є процес передачі даних у комп’ютерних мережах TCP/IP на рівні окремих мережевих потоків та мережевих протоколів.

Предметом дослідження є метод пасивного аналізу мережевого трафіку, технології фільтрації пакетів у просторі ядра, стратегії агрегації та нормалізації метрик якості мережевих з’єднань.

Метою кваліфікаційної роботи магістра є розробка методу моніторингу параметрів комп’ютерних мереж на основі аналізу мережевих, що забезпечує збирання, обробку та візуалізацію метрик якості з’єднання в реальному часі при мінімальному споживанні системних ресурсів.

Для розв’язання поставлених задач використовувалися методи математичного моделювання, методи статистичного аналізу для агрегації метрик, технології фільтрації трафіку в просторі ядра.

Наукова новизна отриманих результатів:

− набув подальшого розвитку метод пасивного моніторингу мережевих потоків, що поєднує фільтрацію заголовків у просторі ядра із статистичною агрегацією метрик у просторі користувача, що дозволяє виконувати діагностику якості з’єднань без копіювання корисного навантаження пакетів та без впливу на мережевий трафік;

− набула подальшого розвитку інформаційна технологія пасивного моніторингу мережевих потоків на основі запропонованого методу, практична

реалізація якого побудована на eBPF та Python, яка інтегрується з існуючими системами збору метрик Prometheus, Grafana та може бути розгорнуте на пристроях з обмеженими ресурсами, зокрема одноплатних комп'ютерах.

На основі проведених досліджень розроблено метод пасивного моніторингу параметрів комп'ютерних мереж із використанням технології eBPF, що забезпечує вимірювання затримок, виявлення повторних передач, оцінку кількості проміжних вузлів та аналіз стану TCP-сесій в реальному часі.

Практична значимість отриманих результатів полягає у полягає у розробленому програмному забезпеченні пасивного моніторингу мережевих потоків eBPF та Python, яке інтегрується з існуючими системами збору метрик Prometheus, Grafana та може бути розгорнуте на пристроях з обмеженими ресурсами, зокрема одноплатних комп'ютерах.

У першому розділі було розглянуто існуючі методи, розглянуто типові проблеми мережевого моніторингу, окремо була розглянута проблема ефективності використання ресурсів.

У другому розділі було визначено типові принцип роботи моніторингу параметрів мережі та архітектуру на які потрібно орієнтуватись щоб забезпечити сумісність та коректність роботи методу із мережевими топологіями.

У третьому розділі було запропоновано метод та алгоритм аналізу параметрів та запропоновано ряд вимог, яких необхідно дотримуватись щоб впровадити та використовувати запропонований метод. Також в цьому розділі було проведено порівняння запропонованого методу із існуючими методами.

У четвертому розділі було проведена фіксація та вимірювання кількості системних ресурсів, які необхідні для коректної роботи методу та порівняння метрик із існуючими методами. В цьому розділі також було зазначено особливості реалізації та проведено ряд експериментів в тому числі із мережевим навантаженням. Також проведене порівняння споживання ресурсів реалізації методу із доступними на ринку інструментами.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ОС - операційна система

ПЗ - програмне забезпечення

BGP - Border Gateway Protocol

SNMP - Simple Network Management Protocol

QoS - Quality of Service

RTT - Round-Trip Time

eBPF - Extended Berkeley Packet Filter

RFC - Request for Comments

DTLS - Datagram Transport Layer Security

ВСТУП

Щоденно кількість бізнесу та користувачів глобальної мережі активно зростають, тому пропорційна ціна похибки та простою мережі. Стабільність мережевого з’єднання та виявлення аномалій стають важливими не лише для бізнесу, а і для конкретного приватного користувача, для якого зниження швидкості, збільшення мережевих затримок чи поява аномалій створюють дискомфорт, а також можуть створювати проблеми для професійної діяльності.

Існуючі системи, очевидно, можуть бути практично застосовувані та виконувати необхідні функції, проте в той же час можуть мати проблемні сторони, бути не надто економними до використання ресурсів або виконувати функції, оперуючи з помірною точністю, цим самим створюючи хибно-позитивні реакції на проаналізовані події.

Недоліки частини готових рішень полягають в тому, що вони не здатні працювати у середовищах, де використання ресурсів та кількість живлення є критичними показниками. Реалізації рішень можуть вимагати значну кількість системних ресурсів, відповідно запускаються лише на промислових серверах, що є недоступним параметром для користувача, який хотів би перевіряти активність та стан власної мережі. При цьому агент сервісу має не заважати самій системі безперебійно оперувати.

Для того, щоб забезпечити оптимальне використання ресурсів і підвищенню ефективності методу і практичній реалізації методу, варто акцентувати увагу на архітектурних рішеннях та оптимізації алгоритму.

Проведений порівняльний аналіз активних та пасивних засобів моніторингу виявив, що пасивний аналіз реального трафіку є більш точним, оскільки не створює додаткового навантаження на канали зв'язку та не спотворює вибірку синтетичними даними. Водночас визначено ключові виклики такого підходу, зокрема необхідність обробки великих обсягів інформації безпосередньо «на льоту» та складність відновлення стану TCP-сесій.

Запропонований метод моніторингу базується на використанні технології eBPF, що дозволяє виконувати фільтрацію та агрегацію метрик безпосередньо у просторі ядра операційної системи. Це вирішує проблему надмірної надлишковості та виснаження ресурсів процесора, притаманну класичним інструментам, які копіюють повні пакети у користувацький простір.

Математична формалізація методу охоплює ідентифікацію потоків за набором ключових параметрів та вимірювання RTT двома способами: через аналіз станів SYN/SYN-ACK та за допомогою опції TCP Timestamp. Такий комбінований підхід забезпечує високу точність вимірювань навіть у складних умовах перевпорядкування пакетів або відсутності підтримки певних опцій мережевим обладнанням.

Впровадження результатів роботи підтверджують, що використання інтелектуальної діагностики аномалій на основі зібраних метрик дозволяє не лише констатувати стан мережі, а й оперативно виявляти причини збоїв, такі як перевантаження буферів чи нестабільність маршрутів. Запропонований підхід забезпечує приріст продуктивності та надійність керування сучасною мережевою інфраструктурою за рахунок пасивного моніторингу параметрів комп’ютерних мереж із використанням технології eBPF, що забезпечує вимірювання затримок (RTT), виявлення повторних передач, оцінку кількості проміжних вузлів та аналіз стану TCP-сесій в реальному часі.

Актуальність роботи полягає в розробці методу моніторингу параметрів комп’ютерної мережі на рівні окремих мережевих потоків. Зростання складності мережевої інфраструктури та обсягів трафіку вимагає ресурсоефективних засобів діагностики, здатних працювати на пристроях з обмеженими ресурсами без впливу на мережевий канал.

Метою кваліфікаційної роботи магістра є розробка методу моніторингу параметрів комп’ютерних мереж на основі аналізу мережевих, що забезпечує збирання, обробку та візуалізацію метрик якості з’єднання в реальному часі при мінімальному споживанні системних ресурсів.

Поставлена мета досягається розв’язанням таких основних завдань:

- проаналізувати існуючі методи та засоби моніторингу параметрів комп'ютерних мереж, визначити їх переваги та недоліки;

- розробити математичну модель ідентифікації мережевих потоків та формалізувати метрики якості з'єднання у вигляді замкненої системи рівнянь;

- розробити метод пасивного моніторингу на основі eBPF-фільтрів з декомпозицією на підсистеми захоплення, обробки та публікації метрик;

- здійснити програмну реалізацію розробленого методу та провести експериментальне дослідження споживання ресурсів і точності вимірювань в умовах реального мережевого навантаження.

Об'єктом дослідження є процес передачі даних у комп'ютерних мережах TCP/IP на рівні окремих мережевих потоків та мережевих протоколів.

Предметом дослідження є метод пасивного аналізу мережевого трафіку, технології фільтрації пакетів у просторі ядра, стратегії агрегації та нормалізації метрик якості мережевих з'єднань.

Наукова новизна отриманих результатів:

- запропоновано метод пасивного моніторингу мережевих потоків, що поєднує фільтрацію заголовків у просторі ядра із статистичною агрегацією метрик у просторі користувача, що дозволяє виконувати діагностику якості з'єднань без копіювання корисного навантаження пакетів та без впливу на мережевий трафік.

Практична значимість отриманих результатів полягає у розробленому програмному забезпеченні пасивного моніторингу мережевих потоків eBPF та Python, яке інтегрується з існуючими системами збору метрик Prometheus, Grafana та може бути розгорнуте на пристроях з обмеженими ресурсами, зокрема одноплатних комп'ютерах.

Для розв'язання поставлених задач використовувалися методи математичного моделювання, методи статистичного аналізу для агрегації метрик, технології фільтрації трафіку в просторі ядра.

За темою кваліфікаційної роботи опубліковано одну публікацію [75] у науковому журналі «Вимірювальна та обчислювальна техніка в технологічних процесах» (Хмельницький – 2026). С. 49-58.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і огляд досліджуваної проблеми

Для початку роботи спрямованої на побудову моделі, необхідно проаналізувати поточні особливості загальної предметної області. Оскільки телекомунікаційні мережі в сучасних реалізаціях суттєво змінили характеристики роботи, відповідно системи моніторингу також почали підлаштовуватись до актуальної проблематики.

Сучасні реалії бізнесу активно проходять цифрову трансформацію, і мережа перестає бути допоміжним інструментом, а її критичність збільшується з кожним роком [1]. Збільшення залежності зумовлено переходом до хмарного обчислення [2], розгортання програмного забезпечення та сервісів на інфраструктурах хмарних провайдерів. Тобто ряд залежних критичних компонентів розташовані віддалено і є ще однією проблемною точкою відмовостійкості [3]. Рух до хмарних рішень є виправданим із точки зору адміністрування, вартості підтримки та оптимізації витрат згідно масштабів використання сервісів хмарного провайдера.

Розглянемо ситуацію, коли дані та сервіси фізично розташовані на хмарних сервісах, в такому випадку будь-яка затримка, деградація мережевого з’єднання може призвести до повної зупинки операційної діяльності бізнесу [4].

Залежність сучасної глобальності бізнесу: ще один вагомий аргумент, коли на розподіленість бізнес процесів, логістику, фінанси, адміністрування діє і залишається критичним параметром можливість швидкої синхронізації в режимі реального часу.

Для маркетингу затримки в розмірі мілісекунд безпосередньо можуть впливати на прибуток або втрату прибутку [5]. Клієнт розраховує на швидкі реакції онлайн-сервісів, затримки у реакції рівноцінно втраті клієнта, відповідно і втраті прибутку.

Також гібридні формати роботи підштовхнули бізнес до рішень, що були не зовсім поширені раніше. Організації, що працюють у індустрії інформаційних технологій почали будувати власні повноцінні корпоративні мережі напряму

пов’язані із корпоративними середовищами працівників [6]. Тому відмовостійкість та стабільність середовищ, в яких організована робота віддалених працівників, з того моменту почала напряму впливати на продуктивність праці [7]. Відповідно стабільність мережі також стала стратегічним фактором у цьому випадку.

Згідно із даними дослідженням “Robert Half” [8] розподіл застосування онлайн форматів варіюється залежно від галузі та становить від 20 до 44 відсотків. Бізнес, що працює у сфері маркетингу, технологій та правничої діяльності найбільше застосовують онлайн, а також гібридний формати роботи.

Діаграма співвідношень форматів роботи у відповідності до галузі подана нижче на рис. 1.1.

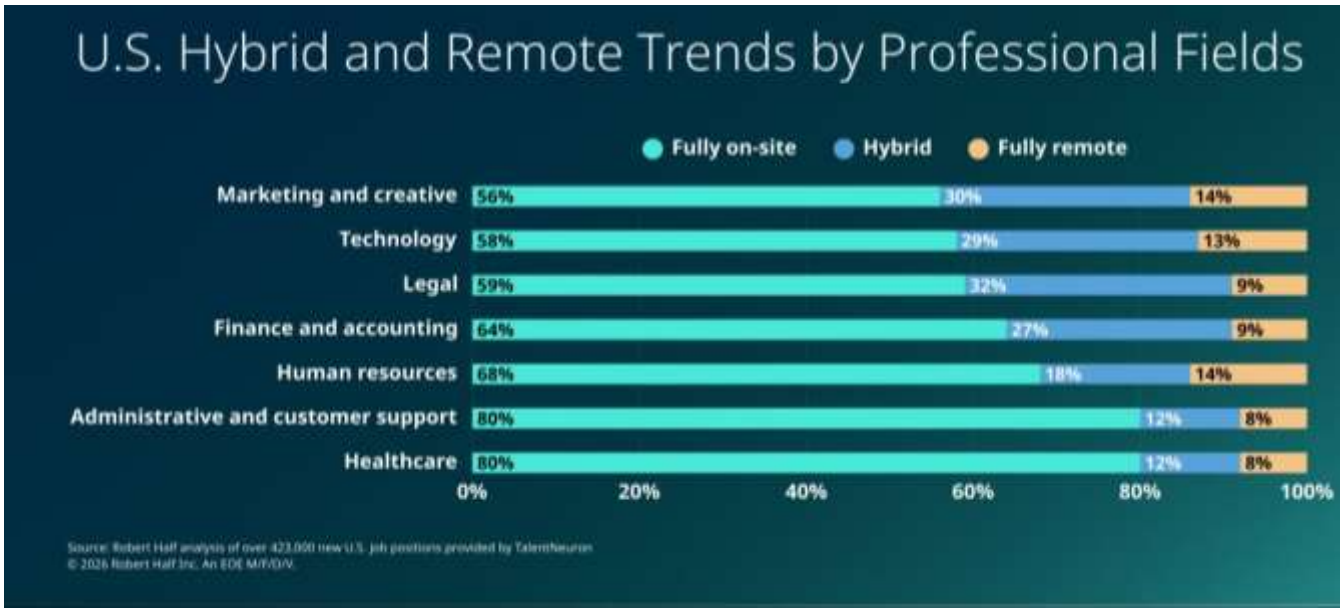


Рисунок 1.1 – Співвідношення форматів роботи відповідно до галузі [8]

Моніторинг якості зв’язку є важливим етапом роботи сервісів та бізнес додатків, моніторинг дозволяє оцінювати і бачити значення, які відображають продуктивність зв’язності [9]. Специфіка бізнесу постійно рухається із гібридного підходу до формату повного переходу на онлайн надання послуг [10]. Тому, навіть коли відсутній повний розрив мережевої взаємодії - це все ще не є точною метрикою, що зв’язок із сервісом є на задовільному рівні [11].

Нестабільність каналу, аномалії маршрутизації, накопичення черг пакетів є факторами, які важко відслідкувати, проте їх критичністю не можна нехтувати.

Існує поняття сірих помилок [12], тобто це стан, коли вузол з'єднання не втрачає повністю зв'язності, а починає працювати, генеруючи помилки і затримки при передачі даних. Основна проблема полягає в тому, що існуючі системи моніторингу можуть вважати подібні аномалії допустимими [13], в той же час аномалії можуть накопичитись за певний час та стати фатальними для бізнес-логіки. Таким чином збільшення затримок мережевих пакетів на 50 мілісекунд може стати першопричиною створення каскадного ефекту [14] і змусить сервіс почати масштабування створюючи додаткові контейнери [15]. При цьому жодних стрибків навантаження не відбувалось, а каскадний ефект виник через проблеми на мережевому або транспортному рівні.

1.2 Аналіз предметної області і огляд досліджуваної проблеми

Класичні системи моніторингу оперують станами "працює" або "не працює", висновок робиться на основі стану виконання одноразового або багаторазового запуску тесту через певний проміжок часу [16], [17].

Критерієм проблеми можна вважати ситуацію, коли розрив між об'єктивним показником роботи системи і досвідом користувачів відрізняється [18]. Звісно, подібні критерії є досить суб'єктивними і вимагають детального аналізу [19], проте коли декілька сотень користувачів спостерігають проблему, а системи моніторинга - ні, це може стати однією із ознак сірої проблеми.

Коли перевіряється стан окремого вузла, аналізується набір метрик конкретного пристрою, це призводить до досить однобокої перевірки стану та може приховати проблему [20].

Класичні системи моніторингу також не можуть оцінити вплив зовнішніх факторів [17] та не можуть ефективно реагувати на каскадні проблеми системи [21]. Тому цим самим під час розслідування проблеми ускладнюють роботу інженерів та можуть створювати хибне враження того, що система працює [22].

Окремо варто розглянути приклади інцидентів у сервісів, які так чи інакше почались із порівняно плаваючої проблеми. Результати аналізу представлені в таблиці 1.1.

Таблиця 1.1 - Аналіз інцидентів, які розпочались із проблем, що не вважались початково критичними

№	Проблема	Першопричини
1	Cloudflare, мерехтлива конфігурація (18 листопада 2025 р.) [23]	Зміна прав доступу бази даних, що збільшила розмір файлу конфігурації вдвічі. Кожних 5 хвилин генерувався новий конфігураційний файл, який передавався всім пристроям по локальній мережі і був помилково інтерпретований як DDoS.
2	AWS сервіси, проблема DNS вирішення (20 жовтня 2025 р.) [24]	У одному із регіонів AWS виникла проблема із іменами домену та їх вирішенням для баз даних DynamoDB. Це спричинило проблему із значною кількістю інших сервісів AWS, проблема проявлялась у вигляді величезних затримок у відповідях сервісів.
3	Microsoft Azure втрата TCP сегментів (29 жовтня 2025р.) [25]	Помилка в конфігурації маршрутизації Azure призвела до некоректного оновлення мережевих правил. Частина TCP сесій розривались ще на етапі встановлення, при цьому ICMP/UDP сервіси працювали як і очікувалось.

Кінець таблиці 1.1

4	Збій Facebook через помилкові налаштування BGP (4 жовтня 2021р.) [26]	Некоректне оновлення конфігурації маршрутизаторів призвело до відкликання BGP маршрутів. Помилка в управлінні створила відсутність маршрутів та повну недоступність сервісів. Без методів глибокого моніторингу відстежити причину проблеми було надзвичайно складно.
5	Витік маршрутів Cloudflare (2019р.) [27]	Аномалія в невеликому мережевому провайдері спричинила затримки в доступі до сервісів, повільне завантаження, оскільки трафік маршрутизувався через вузький канал зв’язку. Інцидент важко було відстежити, оскільки він спостерігався для окремих клієнтів у непостійному вигляді.

Варто зазначити, що системи передачі даних, які використовуються на даний момент, а саме високошвидкісні канали зв’язку, тепер у частині критичних та проблемних станів спостерігають не просто втрату зв’язності з’єднання [28], а у момент проблеми із каналом передачі даних можна спостерігати аномалії [29], які без засобів глибокого аналізу відслідкувати надзвичайно складно [30]. Початкова концепція передбачає, щоб метод і відповідна реалізація системи функціонували у локальній мережі.

1.3 Аналіз загальноприйнятих архітектурних рішень при побудові сервісів

Загальноприйняті архітектурні рішення активно переходять від монолітних сервісів до гнучких та автономних структур. Чималої популярності набула мікросервісна архітектура [31].

Ідея мікросервісної архітектури полягає в тому, що логіка сервісу поділяється на незалежні компоненти [32], що відповідають за окремі функції. Підхід орієнтований на багаторазове перевикористання сервісів та їх взаємодію через шину даних [33].

Також часто архітектори рішень застосовують рішення орієнтовані на події [34]. Великі потоки подій можуть опрацьовуватись брокерами повідомлень в асинхронному режимі. Для ефективності роботи одна частина системи може приймати повідомлення, а інша займатиметься обробкою [35].

Для зручності масштабування, відмовостійкості частини сервісу ізолюється рівнями абстракції та інтерфейсів [36]. Приклад сервісу побудованого на основі мікросервісів поданий нижче на рис. 1.2.

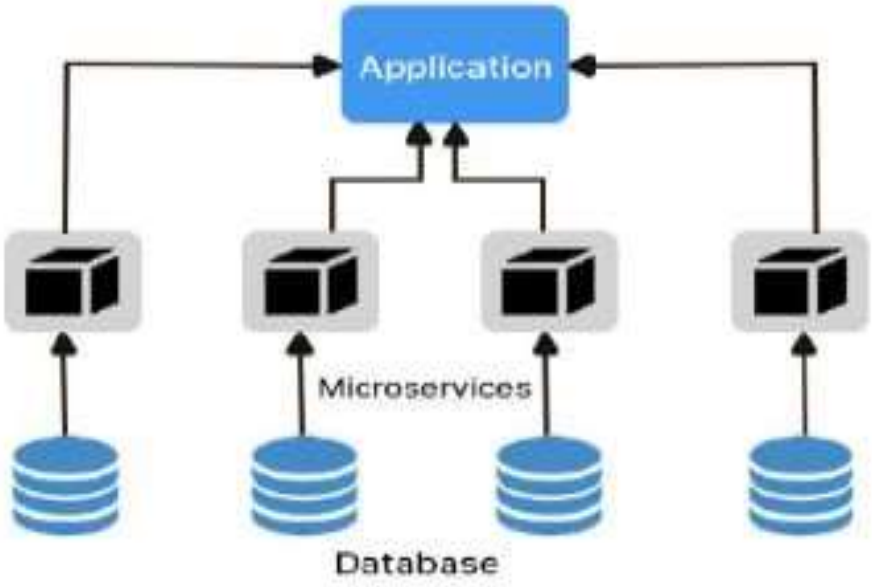


Рисунок 1.2 – Діаграма сервісу побудованого на основі мікросервісної архітектури [37]

Так чи інакше монолітна, мікросервісна, сервісно-орієнтована, безсерверна архітектура, а також рішення орієнтовані на події - реалізовують вимоги бізнесу з точки зору пошуку балансу при розробці [38], можливістю масштабування, бажанням керувати власною інфраструктурою, можливостей фінансів та аспектами регуляторів. Оцінюючи архітектури, не можна обрати єдине ідеальне рішення для всіх можливих вимог [39]. Ідеальне рішення не є чимось уніфікованим та спільним для всіх можливих клієнтів, ідеальним рішенням можна назвати те, що найкраще підійшло під критерії технічного завдання та бізнес-задачу.

Досить часто із засобів контролю використовують SNMP [40], NetFlow [41], smokeping, встановлені на центральному сервері. Їх використання часто включає все активне застосування вертикального масштабування у випадку, коли існуючи розмір сервісу вже не забезпечує стабільну роботу через нестачу системних ресурсів.

Системи із великим географічним розподілом із розташуванням на великих фізичних дистанціях [42] ще більш залежні від методів аналізу якості мережі. При такій архітектурі є зміст не тримати єдиний центральний вузол моніторингу [43], а навпаки розподіляти засоби по регіонах та розділяти їх по проксі-частинах, які збирають метрики. Агрегація та аналіз метрик в свою чергу може відбуватись вже на центральному вузлі [44]. Втрата зв’язності між центральним вузлом і проксі-сервісом - не є абсолютно критичною для всієї системи.

Працюючи над власним методом спрямованим на моніторинг, немає змісту фокусуватись на конкретному архітектурному підході, а з точністю навпаки - намагатись побудувати уніфіковане рішення, щоб збільшити кількість зацікавлених клієнтів.

Звісно, у випадку використання безсерверної архітектури [45], клієнт з високою ймовірністю буде покладатись на можливість моніторингу, що надає хмарний провайдер [46]. Однак для відтворення та тестування сервісу, що відтворює досвід клієнта, окрема система моніторингу має зміст.

1.4 Аналіз існуючих методів та засобів

Якщо аналізувати існуючі реалізації, то ринок рішень є доволі обмеженим та вузькоспеціалізованим [47]. Підприємства, бізнес та індивідуальні користувачі зазвичай інтегрують рішення-агенти у власну інфраструктуру. Акцент уваги зміщується до швидкості реакції, точності, можливостей інтеграції із іншими системами і засобами моніторингу [48].

Актуальні інфраструктурні рішення - це тисячі умовних контейнерів [49], кластерів та уніфікованих сервісів. В той же час із збільшенням розмірів інфраструктури все більше і більше стає актуальною проблема відстеження якості зв’язку, рівні навантаження на канали. Окремо варто розглядати питання, що стосуються безпеки сесій, встановлених від інфраструктури до сервісів, які належать субпідрядникам або організаціям, які відносно попередньо зазначеної інфраструктури є третьою стороною [50].

Інфраструктури великих масштабів, зазвичай побудованої на основі мікросервісів, контейнерів та кластерів, які є фактично ізольовані один від одного [51], проте взаємодіють через визначені інтерфейси.

Також на даний момент все ще існують суцільні монолітні рішення сервісів [52]. У більшості випадків вони існують через певну специфіку реалізації продукту, що вимагає взаємоіснування частин сервісу під керуванням однієї операційної системи. Звісно, даний підхід не є оптимальним та рекомендованим, проте він все ще зустрічається у промислових рішеннях [53].

Тим не менш, і для архітектури, побудованої на основі контейнерів, і для монолітної архітектури, моніторинг якості та стану зв’язності [54] є критично важливим.

Тому промисловість є в активному пошуку рішень, що забезпечують потреби та максимально відповідають параметрам для оптимальної інтеграції у вже існуючі рішення, уникаючи ситуацій, коли доводиться перебудовувати систему кінцевого сервісу задля впровадження рішення моніторингу.

Моніторинг інфраструктури великих розмірів та контейнеризованої інфраструктури значно відрізняється від класичних підходів [55] через високу динамічність і короткий час виконання об’єктів, над якими планується проводити спостереження. Автомасштабування сервісів може створювати умови, коли декілька сотень нових контейнерів будуть створені всього на декілька секунд або хвилин саме у момент збільшення навантаження, а після зниження навантаження будуть знищені, оскільки вони вже виконали поставлену на них функцію - швидко опрацювали скачок навантаження [56].

Також для моніторингу великих інфраструктур існує проблема, що стосується бар’єру видимості, що полягає в рівнях інкапсуляції між основною операційною системою та контейнерами [57]. Проблема, яку створює інкапсуляція для моніторингу полягає в тому, що агенти моніторингу є не зовсім ефективними. В той же час додавання контейнерів спрямованих на вирішення проблем моніторингу, поряд із сервісом є не зовсім ефективним методом використання ресурсів.

Варто також розглянути засоби, які проводять опитування системи як окрему складову моніторингу. Дана модель передбачає періодичне опитування сервісів, тобто система сама ініціює збір метрик через певний проміжок часу [58]. На основі зібраних метрик система опитування може провести порівняння із пороговими значеннями та визначити стан системи [59].

Складова моніторингу, що базується на самостійній генерації повідомлень із точністю навпаки, змушує сам сервіс проводити самоперевірку та повідомляти центральну систему моніторингу про свій аномальний або критичний стан. Із часом цей підхід проявив свої слабкі сторони. [60] Досить надлишковою є реалізація перевірки стану в кінцевому сервісі.

Існуючі рішення досить часто є побудовані на основі сервісів-агентів, що встановлюються на пристрій та можуть активно запускати синтетичні тести, аналізуючи їх результати. Агент відправляє певну кількість пакетів та аналізує кількість отриманих відповідей.

Тим не менш, синтетичні тести [61] можуть не виявити аномалій, так як вони запускаються у відносно контрольованих і спрощених умовах, не враховуючи

змінних факторів та непередбачуваних умов. Трафік може бути неоднорідним, відповідно його опрацювання теж може бути випадковим і неоднорідним між початковим і кінцевим вузлами.

Щодо рішень доступних на ринку, то одним із найпопулярніших інструментів пасивного аналізу є Wireshark [62], однак він не є автономним і інтуїтивним. Wireshark дозволяє глибоко аналізувати пакети, їх структуру, але призначений суто для ручного аналізу і діагностики. Він не призначений для тривалого моніторингу великих мереж.

Для більш постійного моніторингу використовується інструмент Zeek [63]. Він аналізує трафік і створює журнал подій із запитів, сесій і з'єднань, а також підозрілої активності. Це рішення вимагає досить великого об'єму ресурсів і для початку роботи необхідне складне налаштування.

Існують також інстурументи орієнтовані на потоки трафіку – ntopng [64], SolarWinds NetFlow Traffic Analyzer [65]. Ці інструменти орієнтовані на аналіз потоків трафіку і статистику використання мережі. З точки зору візуалізації вони мають чудовий інтерфейс із інтуїтивним керуванням. Однак аналіз потоків за своєю суттю передбачає обмеження [66]. Системи не можуть відстежувати затримки, втрати, аномалії маршрутизації, оскільки протокол Netflow цього не передбачає [67].

1.5 Постановка задачі та аспекти, на які варто акцентувати увагу

Якщо розглядати проблематику, яку має вирішувати система моніторингу, то одним із важливих етапів є постановка задачі. Чітке визначення ключових моментів при постановці задачі є критично необхідним, при будь-якому проєктуванні.

На початкових етапах важливо визначити цілі та очікуваний кінцевий результат, так як можливості реалізації є доволі широкі, то концентрація на ключових елементах має зміст [68]. Раціональний розподіл ресурсів - це не менша перевага системи.

При реалізації варто зосередити увагу на пріоритетах, оскільки зазвичай користувач очікує побачити результат, що вирішує конкретну проблему, а не продукт розпорошений на множину функцій, які не є повністю необхідними.

Визначення чітких критеріїв [69] на початку проєктування спрямоване на зменшення ризиків та допомагає виявити і виправити системні помилки [70] на початкових етапах, тому і знижує потребу в додаткових корегуваннях або правках в архітектурі рішення.

Ще одним важливим етапом є побудова плану із ключових точок. Ключові точки спрямовані на моніторинг та відстеження прогресу для проміжного контролю. Так чи інакше, можуть виникнути ситуації, коли частину функцій доведеться уніфікувати або ре-інтегрувати з метою оптимізації.

У розрізі системи постановка задачі є ще більш важливою, ранні етапи визначають чи система справді допоможе виявити аномалії та допоможе в розслідуванні короткотривалих проблем, чи навпаки лише додасть шуму в моніторингу, створюючи велику кількість хибних спрацювань.

Якщо оцінити засоби, що згадувались в попередньому розділі, то досить часто реалізації зіштовхуються із проблемою мінливої працездатності. Фактично класичні засоби і рішення орієнтуються на стан інтерфейсів чи каналу передачі даних в цілому. Підхід “працює чи не працює” в цілому, не завжди може виявити проблему деградації якості з'єднання. Також існуючі рішення часто використовують більш синтетичний підхід, коли сама ж система моніторингу генерує потоки трафіку ICMP або UDP, який може бути не зовсім корисним для розбору проблеми. Згенерований синтетичний трафік може не відобразити проблему датаграм, хоча проблема для сегментів TCP може продовжувати існувати паралельно.

Тобто генерування потоків трафіку не завжди демонструє аналогічну поведінку справжнього трафіку сервісів, так як механізм комутації пакетів може передбачати маршрутизацію через випадкові вузли мережі.

Окремо варто звернути увагу на кількість системних ресурсів, що використовуються для моніторингу. Досить часто існуючі системи є вимогливими з точки зору процесорного часу, використовують значну кількість оперативної та

постійної пам’яті. Це пов’язано із великою кількістю метрик, що збираються та опрацьовуються. На використання оперативної пам’яті може впливати механізм кешування історичних даних - індексів бази даних, тимчасових метрик.

Постійна пам’ять використовується теж досить активно, запис історичних даних спричиняє окреме навантаження на дискові підсистеми. Навантаження на мережу спричинене передачею телеметрії, за умови, що метрики не опрацьовуються локально.

Досить часто рішення повністю будуються на інтерпретованих мовах програмування - Python, Perl, Ruby. Це спрощує логіку системи моніторингу, але при цьому погіршує швидкість її роботи, використовує надлишкові ресурси для інтерпретації до машинного коду. Компільовані мови програмування є значно кращими в межах обробки коду та керування ресурсами, але в той же час можуть ускладнювати логічне представлення сервісу.

При порівнянні швидкості обробки коду написаного на C та Python було практично оцінено, що швидкість обробки коду на Python значно повільніша ніж на C/C++, результати тестів подані на рис. 1.3. Тому переваги обробки C/C++ порівняно із Python будуть особливо помітні на обладнанні із малою кількістю ресурсів. В свою чергу Python має іншу перевагу, це якість, інтуїтивність, простота коду. Для пристроїв, де наявна мінімальна кількість ресурсів, може бути доступний MicroPython.

MicroPython є більш спрощеною і оптимізованою реалізацією мови Python, що застосовується на мікроконтролерах, але водночас може бути розгорнута в середовищах, де доступно значно більше системних ресурсів. Кількість модулів є зменшена, існуючі модулі MicroPython можуть містити спрощений набір функцій.

Так чи інакше, існують суттєва різниця між Python та MicroPython, способи завантаження, керування пам’яттю, можливості для роботи із системами різного масштабу. Але рішення скороченого набору функцій може бути виправданим при вирішенні проблем оптимізацій для систем із мінімальною кількістю ресурсів.

Для балансу продуктивності та простоти є зміст розглядати комбіноване рішення із двох мов програмування в єдиному проєкті.

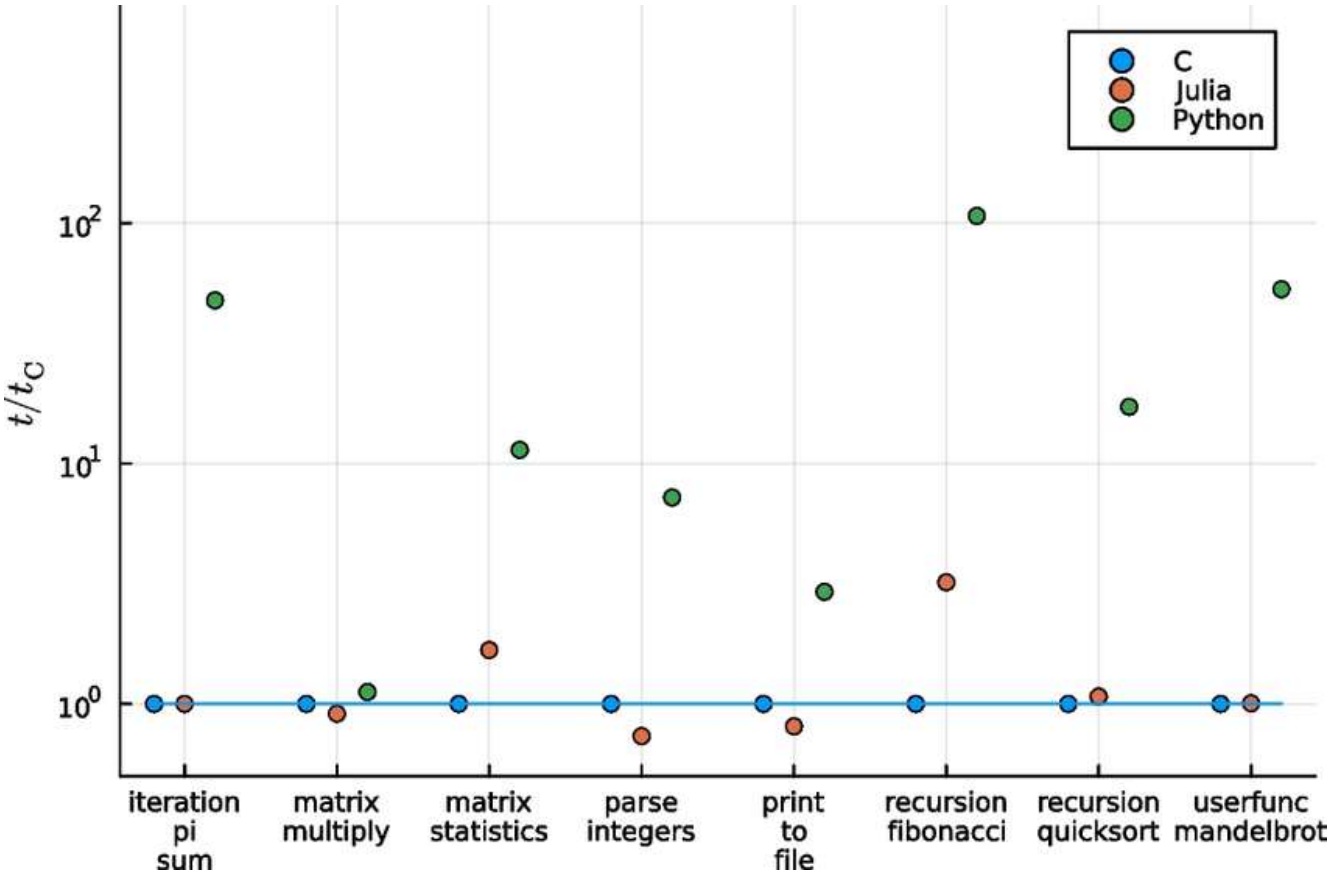


Рисунок 1.3 – Порівняння швидкості обробки мов програмування C та Python [71]

Таким чином, можна визначити основні вимоги, з яких і формується технічне завдання. Система має бути сфокусована на аналізі пакетів мережі, що надіслані сервісами. Сам аналіз і фільтрацію варто проводити якнайбільш низькорівнево і якнайдалі від користувацького простору. Система має працювати використовуючи мінімальну кількість ресурсів. Попередня вимога має дозволити запуск на одноплатних комп'ютерах відповідно. Щоб зменшити кількість втрат на інтерпретації мов програмування та не втратити можливості у простоті сервісу, варто застосувати комбінацію із інтерпретованої та компільованої мов програмування. Це дозволить розділити логіку ресурсоємкої фільтрації та частину, що відповідає за керування і візуалізацію.

Також для уніфікованого розгортування на різних видах системних архітектур можна звернути увагу на використання технології контейнеризації - Docker. Це спростить процес встановлення і дозволить отримати робочу систему моніторингу виконавши декілька команд.

Також система має працювати точно та не створювати надмірних хибних спрацювань, забезпечуючи достатню глибину аналізу. Логіка має працювати таким чином, щоб мінімальні зміни мережі могли бути помічені, а не визначати лише загальні стани: “мережа працює або повністю не працює”. Детальний аналіз TCP сегментів має допомогти із вирішенням сірих проблем.

В реаліях, де активно набувають популярності пристрої IoT, для будь-яких завдань, агрегований моніторинг з'єднання може бути не менш актуальним. Велика кількість пристроїв, кластери контролерів, не дозволяють розгорнути складне, ресурсоємке рішення, але при цьому покладаються на віддалені вузли для великих обчислень, зберігання вимірів, контролю, керування ще більше, ніж класичні сервіси. Стан мережі, якість з’єднання - це ті критичні показники, які можуть повпливати на чимало факторів.

Постановка задачі зводиться до фундаментальної проблеми та пошуку рішень. Інструменти моніторингу можуть бачити результат, що сайт відкривається повільно, або швидкість завантаження є низькою, але причина такої поведінки є невідомою для інструментів, що оперують на рівні 7 моделі OSI. Диференціація причин аномальної поведінки стає перевагою системи.

Відповідно етапом вирішення проблеми стає декомпозиція технічного завдання на етапи моніторингу: пасивний аналіз, перевірка та глибокий аналіз TCP сегментів, визначення діагностичних станів, врахування обмежень ресурсів середовища, уніфікація розгортування моніторингу.

1.6 Висновки до першого розділу

У межах розділу 1, було проведено аналіз існуючих методів і засобів оцінювання стану мережі. На основі проаналізованого матеріалу можна зробити висновок, що сучасні підходи до моніторингу базуються на поєднанні активних і пасивних методів збору даних, використанні централізованих і розподілених систем. Існуючі методи значним чином варіюються у підходах для збору метрик.

Обчислення метрик у існуючих реалізаціях значним чином відрізняється. Активні методи дозволяють оцінити потенційні характеристики мережі за допомогою штучного трафіку, проте не завжди відображають реальну поведінку системи в умовах змінного навантаження. Пасивні підходи, навпаки, дають більш об’єктивні значення, що демонструють роботу мережі, але потребують значних обчислювальних ресурсів, складної інфраструктури для обробки великих обсягів телеметрії.

Оскільки основна частина методів, що працюють пасивним чином потребують чималих ресурсів, то поставлене завдання передбачає оптимізацію споживання ресурсів завдяки оптимізації методу та зміні архітектурного стеку, порівняно із існуючими методами та реалізаціями побудованими на основі них.

Загальні архітектурні рішення значно варіюються, відповідно можливості інтеграції методу в існуючу архітектуру також були визначені як частина технічного завдання.

2 МОДЕЛЬ СИСТЕМ МОНІТОРИНГУ ПАРАМЕТРІВ КОМП'ЮТЕРНОЇ МЕРЕЖІ

2.1 Принцип роботи моніторингу параметрів мережі

В цілому мережевий моніторинг із розвитком IT-інфраструктури став постійною складовою керування й адміністрування інфраструктури в цілому. Загальне поняття моніторингу враховує аспекти періодичного або постійного збирання, аналізу, обробки метрик про стан взаємодії пристроїв, сервісів із ресурсами або пристроями, що розташовані із ними у одній і тій же локальній мережі, або що розташовані віддалено із доступом через випадкові проміжні вузли глобальних мереж. Поняття фіксованого рівня обслуговування QoS порівняно не часто використовується для типових користувацьких мереж та мереж дрібного бізнесу, але для мереж із великими кількостями трафіку поняття є актуальним.

Стандарт ISO/IEC 27033 орієнтований на аспекти безпеки при моніторингу, проте ефективна система моніторингу забезпечує опрацювання подій в реальному часі, журналювання, збереження метрик та засоби сповіщення адміністратора.

Засоби, спрямовані на мережеву телеметрію, було поділено на категорії, класифікуючи їх за способом збирання метрик. Активні і пасивні засоби напряму відрізняються способами.

Розглянемо завдання систем моніторингу із векторної точки зору. Нехай F={f1,f2,...,fn} - множина TCP-потоків, де кожен потік fn визначається чотирма параметрами IP адреси ініціатора, IP призначення, мережевий порт ініціатора та порт призначення, src_ip, dst_ip, src_port, dst_port.

Для кожного fn визначено вектор характеристик, який представлено формулою:

$$Q(fn) = \{RTT, \sigma RTT, L, W, H\}, \quad (2.1)$$

де RTT – затримка;

σRTT – різниця між затримками;

L – рівень втрат пакетів;

W – розмір вікна прийому;

H – кількість проміжних вузлів.

Активні методи генерують тестовий трафік, окремі потоки, на основі яких можна виміряти параметри мережевого з'єднання на шляху до точки призначення. Активні засоби створюють синтетичний трафік.

Множина усіх активних потоків спостереження утворює скінченну множину, що подано у вигляді формули:

$$F = \{f1, f2, \ldots, fN\}, \; F = N, \tag{2.2}$$

де F – множина потоків;

N – потужність множини, тобто кількість активних потоків в інтервалі спостереження.

Рівень втрат пакетів у потоці визначається як співвідношення, представлене формулою:

$$L = P\_lost \; / \; P\_sent, \tag{2.3}$$

де P_lost – кількість втрачених пакетів;

P_sent – загальна кількість надісланих пакетів.

Активно використовувались засоби для ICMP-тестування, використовуючи утиліти ping, traceroute(частина реалізацій утиліти застосовує саме ICMP). Для визначення маршруту застосовуються traceroute і mtr. Щоб виміряти пропускну здатність застосовується iperf3. Окремо визначено засоби тестування, що починають свою роботу із прикладного рівня, тим не менш в кінцевому результаті тести прикладного рівня все одно відображають більш низькорівневе тестування на 2-3-4 рівнях моделі OSI.

Активний метод аналізу з'єднання має ряд переваг і недоліків. Щодо переваг, то тестування можна провести у довільний час, адаптуючи вхідні параметри так,

щоб результати були найбільш показовими. Водночас недоліком є те, що трафік є повністю синтетичним і за певних умов не показує проблему, яка існує для справжнього трафіку системи. Таке вимірювання характеризує лише поведінку синтетичного трафіку. Окрім цього, згенеровані окремі потоки – це додатковий трафік, що циркулює мережею, великі об’єми трафіку при тестах iperf3 завантажують канал і напряму негативно впливають на справжній трафік користувача. Також синтетичний трафік піддається блокуванню від мережевих екранів і засобів вторгнення.

Пасивні засоби спрямовані на аналіз існуючого трафіку, не генеруючи нового. Глибокий аналіз заголовків пакетів дав розуміння процесів, що відбуваються в мережі на основі пакетів, що прибули до пристрою. Аналіз концентруєтеся на трафікові застосунків і сервісів. Пасивні засоби не впливали на канал зв’язку та не спотворювали вибірку потоків синтетичними даними. Вплив мережевих екранів та аналізу також не притаманний для цього способу. Трафік, що циркулює від пристрою до сервісу, який розташований на сервері, має бути дозволеним. Якщо сервер не готовий приймати запити, у випадку встановленої TCP сесії, її буде не помітно, у випадку UDP/ICMP – не видно відповідей.

Формалізуємо роботу пасивного аналізу. Поєднання пакета у відповідність із потоком представлений функцією:

$$g(p) = src\_ip(p), dst\_ip(p), src\_port(p), dst\_port(p), \quad (2.4)$$

де p – пакет який аналізується;

$g(p)$ – функція, що повертає чотирискладовий ключ потоку, до якого пакет належить;

src_ip – вихідна IP адреса;

dst_ip – IP адреса призначення;

src_port – вихідний порт;

dst_port – порт призначення.

Окремо визначаються гібридні засоби. Гібридні засоби моніторингу застосовують і активні, і пасивні тестування мережі. Коли пасивний аналіз зафіксував аномалію, активний аналіз допомагає локалізувати проблему, не створюючи надмірне навантаження.

Підвищення рівнів складності в рішеннях мережевої інфраструктури відповідно адаптує задачу моніторингу для контролю за мережевими з'єднаннями і якістю передачі даних. Більш традиційні підходи, що використовують як основні метрики – метрики операційної системи: завантаження каналу мережі, завантаження процесора, кількість вільної оперативної пам'яті, проявляють недолік у вигляді статичності. Особливість полягає в тому, що стан каналу і мережі в цілому для різних потоків даних може бути різний. Це класична особливість підходу комутації пакетів при передачі даних із точки А в точку Б.

Об'єктом дослідження систем моніторингу є процес передачі даних у мережах TCP/IP на рівні окремих мережевих потоків. Мережевий потік - це послідовність пакетів, у випадку TCP, окремі сегменти корелюються в єдиний потік за допомогою номера послідовності, IP адреси джерела й призначення та портами джерела і призначення.

2.2 Архітектура систем моніторингу

Архітектура систем моніторингу є досить багатогранною і багаторівневою. Її умовно поділяють на рівень зберігання та представлення результатів, рівень обробки і нормалізації та рівень збирання метрик.

Рівень зберігання та представлення результатів спрямований на те, щоб забезпечувати тривале збереження попередньо зібраних метрик у часових рядах. На цьому ж рівні також відбувається візуалізація. Цей рівень напряму взаємодіє із інструментами, що широко застосовуються в промисловості, типу Prometheus та Grafana, InfluxDB, Elasticsearch, New Relic. Існуючим рішенням часто не вистачає можливостей інтеграції в існуючі системи моніторингу для існуючої інфраструктури.

Рівень обробки та нормалізації здійснює обробку і класифікацію метрик, на цьому рівні відбувається обчислення показників та виявлення відхилень у параметрах роботи комп'ютерної мережі. Цей рівень також передбачає порівняння отриманих метрик із еталонними або із середнім значенням.

Рівень збирання метрик реалізований різними механізмами. Реалізація може бути побудована на активному, пасивному або гібридному зборі метрик. Існуючі реалізації систем моніторингу є переважно активного формату, на основі агентів.

Щодо реалізацій архітектури, то архітектура будується у централізованому форматі, коли всі із рівнів розгорнуті на одному пристрої. Це спрощує управління, але погіршує відмовостійкість та ускладнює можливості масштабування.

Абсолютна протилежність централізованої архітектури – розподілена. Розподілена система будується на основі окремих агентів, встановлених на вузли мережі. В свою чергу агрегація і зберігання метрик відбувається на окремих серверах. Розподілена архітектура дає можливість збирати метрики в безпосередній наближеності до пристрою, комунікацію якого потрібно відстежувати. Водночас розподілена архітектура породжує проблему надлишковості, коли на множині проміжних вузлів через які проходить трафік, при рухові трафіку до шлюзу за замовчуванням або орієнтуючись на маршрутизацію, отримується віддзеркалення одного і того ж трафіку в різних місцях, тобто відбудеться дублювання.

Розподілена система має ряд переваг для масштабування, але доцільність розгортування агентів має оцінюватись окремо, в залежності від існуючої архітектури, топології мережі і потреб окремого клієнта.

Архітектурні рішення зазвичай враховують впровадження механізмів синхронізації та забезпечують нормалізацію і узгодження зібраних даних.

Візуалізація багаторівневої архітектури системи моніторингу гібридного типу зображено на рис. 2.1. На рисунку відображені Prometheus і Grafana як засоби агрегації та візуалізації, проте замість них можуть використовуватись і інші альтернативні рішення. Для кращої візуалізації та можливості порівняння, було зображено методи, які потребують агента та не потребують його.

Використавши агента для активних запитів та не агентну систему, до використання став доступний гібридний метод.

Рисунок 2.1 - Багаторівнева архітектура системи моніторингу гібридного типу

2.3 Принципи обробки трафіку системами моніторингу, що базуються на пасивному аналізі

Оскільки моніторинг цього типу базується на спостереженні за справжнім трафіком, що передається по каналу зв’язку, то система переважно працює як окремий сегмент отримувача.

Для аналізу застосовуються або повні копії кадрів, пакетів, сегментів і датаграм, або лише їх заголовки із ключовими полями. На відмінну від систем протилежного типу, де аналізуються синтетичні дані, в системі сегменті-отримувача з’являється ускладнення – є потреба аналізувати потік трафіку в реальному часі, не маючи можливості виконати повторний запит.

Коефіцієнт скорочення обсягу даних, що передані у простір користувача, поданий формулою:

$$r = S\_header / S\_packet, \quad (2.5)$$

де r – коефіцієнт скорочення;

S_header – розмір заголовка, що передається;

S_packet – повний розмір пакета.

Іншою проблемою, що потребує вирішення, є обмежений контекст. Проаналізувати TCP сесію, без аналізу етапу встановлення і трьох етапної комунікації досить складно.

Система отримує потік фрагментів, що не відносяться один до одного, в довільній послідовності. Якщо аналізувати транспортний рівень моделі OSI сегменти, датаграми несуть лише поля заголовку: порядковий номер, розмір вікна прийому, часові мітки, у випадку TCP – стани сесії. Жодне із цих полів не дозволило одразу зробити висновок про стан з'єднання. Відповідно система, що аналізує справжні пакети, є автоматом відстеження стану з'єднань та передбачає механізм поєднання пакетів за ознаками, щоб встановити їх спорідненість. Вигляд формату TCP заголовка подано на рис. 2.2.

Переважна більшість існуючих методів застосовують повне копіювання пакетів із простору ядра операційної системи в простір користувача із корисним навантаженням. Пакет потрапляє через мережевий інтерфейс в буфер, інструменти типу Wireshark, Zeek досить часто застосовують цей спосіб. Спосіб має чимало готових до використання бібліотек та є досить універсальним в розрізі використання для систем моніторингу. Проте містить значні недоліки, одним із яких є надмірна надлишковість.

Надлишковість негативним чином впливає на результати спостереження, оскільки навіть мінімальна втрата ресурсів в умовах одноплатних комп'ютерів також впливає на результат та його точність. Збільшення похибки через невдале технічне рішення значно погіршує характеристики методу у порівняння із аналогами. Тому задля більшої доцільності було враховувати фактори, оптимізація яких зберігала ресурси.

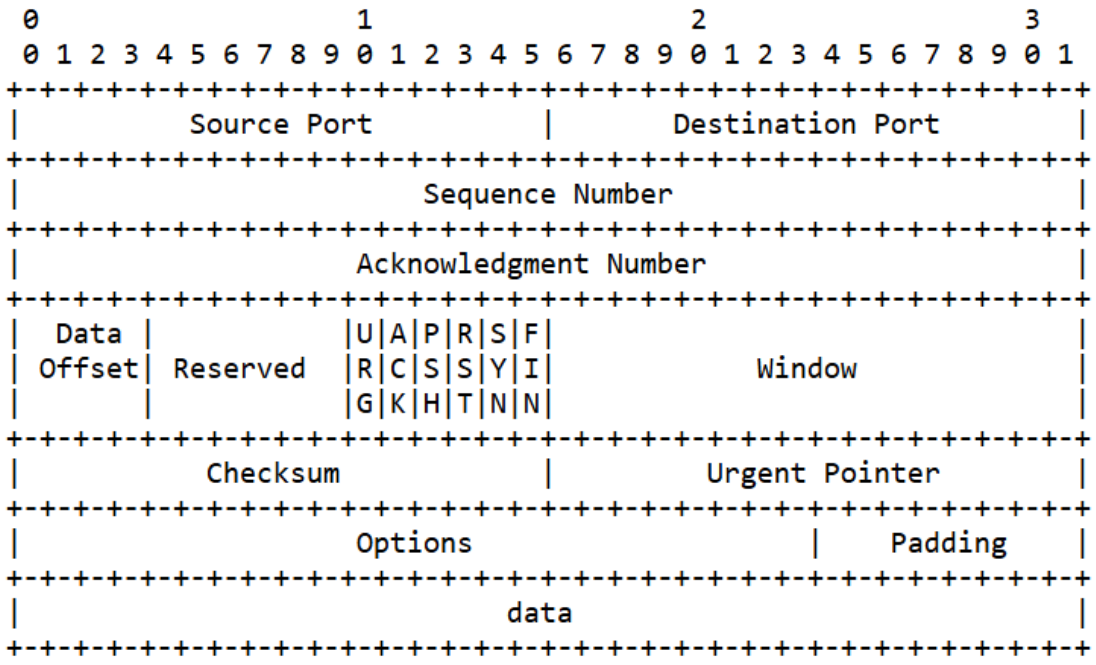


Рисунок 2.2 – Вигляд формату TCP заголовка [72]

Для того, щоб оцінити стан мережі, системі моніторингу не потрібно аналізувати поля із інформацією (поле data в заголовку). Іншим недоліком є копіювання пакету в користувацький простір, при рості використання пропускної здатності, система аналізу із невеликою кількість ресурсів не зможе коректно працювати через виснаження ресурсів процесора, часу використання ядер і оперативної пам’яті. Також при надсиланні надмірної кількості трафіку і виснаження ресурсів, операційна система за стандартним механізмом роботи може розпочати відкидання пакетів. Це одна із причин, коли результати вимірів спотворюються.

Можливе також застосування технології TAP, це фізичний пристрій, що дублює трафік на фізичний інтерфейс, звідки трафік можна зняти для подальшої обробки. Це апаратний пристрій, який здатен захоплювати і передавати кадри із високим рівнем точності. Головні недоліки - фізичне втручання в інфраструктуру, встановлення в кожному сегменті мережі, складність масштабування і створення додаткової точки відмови. Ця технологія підходить для аналізу мережевих загроз та засобів виявлення вторгнень, однак надмірна надлишковість не дає можливості застосувати технологію із пристроями, що мають мінімальну кількість ресурсів.

Із існуючих способів також існують протоколи типу NetFlow, протоколи агрегованої телеметрії. За допомогою цього методу отримуються одразу агреговані потоки за класифікаціями, водночас загальна статистика не є надто детальною і її не є достатньо для аналізу параметрів і стану мережі. За допомогою стандартного NetFlow важко визначити аномалії канального і мережевого рівнів.

Порівняно недооціненим методом є захоплення і фільтрація заголовків за допомогою фільтрів eBPF (extended Berkeley Packet Filter). Це код, що виконується напряму у просторі ядра без переведення пакетів в користувацький простір для аналізу. Аналіз трафіку відбувається без виходу за межі простору ядра, а в користувацький простір повертаються лише дані, які були відфільтровані. Розмір пакету зменшується в декілька десятків разів, можливість обробки корисних даних збільшується відповідно. Основні складнощі при створенні реалізацій - це чітко визначений синтаксис, глибокий аналіз логіки, верифікація перед запуском, щоб уникнути неочікуваних умов в коді. Це складність, яка з’являється для реалізацій, але однозначно не недолік, це засіб захисту системи і спосіб забезпечити стабільну роботу операційної системи, недопустивши неочікувані умови до виконання.

Обмеження eBPF полягають в тому, що існує залежність від версії ядра Linux 4.9 (дата офіційного виходу грудень 2016 року), на більш старіших версіях були впровадженні певні функції eBPF, але частина із функцій додавались із оновленнями до версії 4.9.

Незалежно від застосованих технологій та специфік тих чи інших реалізацій, основне завдання у випадку відстеження TCP сесій зводиться до відновлення стану TCP-сесій та розбір і пошук спорідненості датаграм UDP із великого потоку даних. Протокол TCP є з’єднано орієнтованим і збір метрик має місце в контексті конкретної сесії. Система починає працювати із вже встановленими з’єднаннями, без видимості трьох етапного рукостискання, відповідно такі сесії треба обробляти окремо за іншим механізмом, щоб не створювати хибних метрик.

Глобальні мережі не є ідеальним середовищем із ідеальною точністю передачі даних, відповідно прибування пакетів івдбувається не в початковій

послідовності, а в перевпорядкованому форматі. Це прийнятна поведінка мережі, однак і рішення моніторингу потрібно адаптовувати відповідно.

Також система спостерігає за сигналами FIN та RST і проактивно очищає записи про завершення сесій.

Попередньо зазначались особливості опрацювання TCP потоків трафіку. Однак існує не менш важливий транспортний протокол UDP, що працює без встановлення з'єднання і немає механізму підтвердження доставки датаграми.

Через специфіку роботи протоколу моніторинг будується із іншим підходом для збирання телеметрії, в цілому моніторинг є більше обмежений в кількості корисних даних, на основі яких робляться висновки про роботу мережі.

Кількість UDP датаграм в мережах та їх функції є значними. Велика кількість протоколів використовують саме UDP на транспортному рівні, щоб зменшити затримки при передачі даних за рахунок уникання повторного надсилання даних та відсутності механізму підтвердження. Вигляд заголовку UDP подано на рис 2.3.

Для UDP датаграм визначена корисна інформація – це інтенсивність потоку із пропускною здатністю, міжпакетний інтервал. Та однією із найбільш цінних метрик є рівень втрат і зміна послідовність датаграм, що визначена із порядкового номера пакету.

Вимірювання RTT значень для UDP є досить нестандартним завданням, враховуючи відсутність стандартного механізму відповіді на транспортному рівні.

У випадках аналізу трафіку протоколів DNS, NTP, RADIUS на прикладному рівні отримуються більше корисних деталей, приміром ідентифікатор транзакції.

Використання комбінованих даних із транспортного рівня в парі із даними зібраними на прикладному рівні дає більш детальний результат. Пов'язавши запит-відповідь до транзакції, оцінюється стан мережі і проміжних пристроїв, через які пройшов UDP трафік до точки призначення.

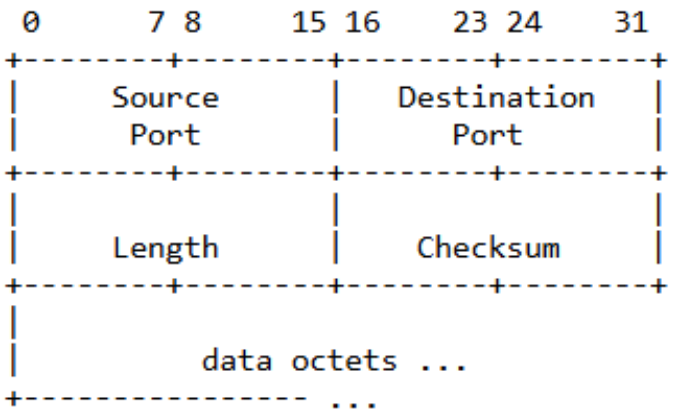


Рисунок 2.3 – Вигляд заголовка UDP протоколу [73]

Значного поширення набув протокол DTLS, призначений для шифрування датаграм. Шифрування датаграми не дозволило проаналізувати вкладення вище мережевого рівня. Лише доступний заголовок IP рівня, адреси, порти, розмір пакетів і часові мітки. Набір інформації є досить обмеженим, однак все ще була можливість визначити пропускну здатність і різницю затримок між пакетами.

У випадку DTLS завдання повністю зводилось до класифікації застосунків і пошуку аномалій без можливості аналізу і розшифрування вмісту датаграми і даних енкапсульованих на вищому рівні. Існує декілька версій DTLS, різниця між DTLS 1.2 і DTLS 1.3 подана на рис. 2.4.

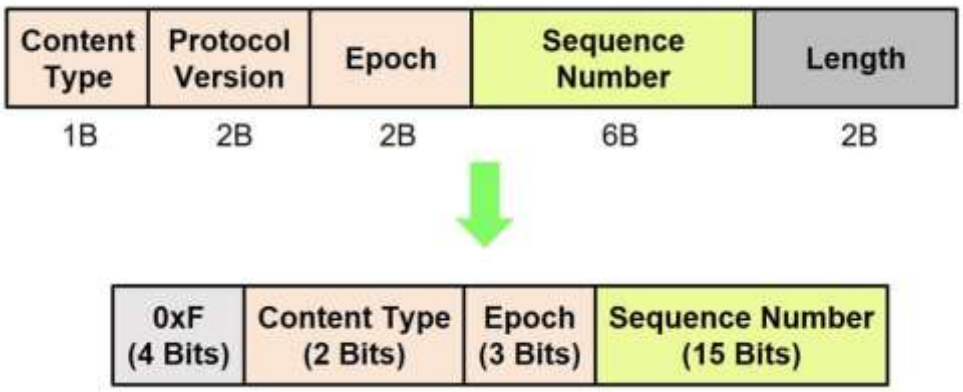


Рисунок 2.4 – Вигляд заголовка DTLS протоколу та різниця між версіями DTLS 1.2 і 1.3 [74]

2.4 Вибір стратегії агрегації метрик в часі

Пасивний моніторинг мережевого трафіку генерує практично безперервний потік трафіку. Зберігання і передача абсолютно всіх метрик як частини спостережень є технічно складним, ресурсоємким завданням. Системи зберігання метрик також мають ліміт пропускної можливості для читання запису сховища.

Попередньо зазначена проблема породжує інше технічне завдання, що розв’язана на рівні архітектури. Оскільки нарощування системних ресурсів не є оптимальним вирішенням проблеми, то створення рівня агрегації стає ще більше актуальним. Завдяки агрегації проміжні результати адаптовувались статистично, цим самим спрощувалось завдання зберігання, фіксацій та передачі.

Вибір стратегії агрегації напряму впливає на швидкодію системи моніторингу, можливості реагування на аномалії, точність, обчислювальну складність.

Основні класи метрик були поділені на два типи - лічильник та вибірку. У вигляді лічильнику подаються зростаючі величини: кількість трафіку, байтів, кількість пакетів, що прибули в довільному порядку. Найбільш оптимальним способом для таких величин є агрегація за інтервал часу.

Для значень таких як RTT, розмірів вікна прийому в заголовку, лічильник для агрегації не є не інформативним. Кращу оптимальність проявили статистичні характеристики розподілу в розрізі часового проміжку: середнє значення, медіана, стандартне відхилення.

Вся тривалість дослідження умовно розділялась на інтервали і на основі зібраних метрик в інтервалі був створений агрегований запис. Кожного інтервалу проводився запис зібраних метрик. Така методологія дала можливість створити більш передбачувану поведінку за різних вхідних умов. Таким способом було отримано усереднений вимір за інтервал. Однак інтервальний поділ також продемонстрував недолік, за умови, що аномалія розпочалась і закінчилась у момент переходу між інтервалами.

Альтернативним методом є спосіб із застосуванням ковзного вікна, коли в пам’яті відбувається збереження фіксованої кількості метрик, а при досягненні максимальної кількості найстаріший запис витісняється новим. Щільність вимірювань збільшується. При зменшенні трафіку спостерігався недолік, метод ковзного вікна продемонстрував повільну поведінку.

Для систем із обмеженою кількістю ресурсів та задачею, коли точність і швидкість спрацювання важлива, було розглянуто метод експоненційно зваженого із ковзним середнім. В межах цієї стратегії зберігається єдине агреговане значення, яке оновлюється при надходженні нового, подане у вигляді формули:

$$S_{\mathrm{n}} = \alpha * x_{\mathrm{n}} + (1 - \alpha) * S_{\mathrm{n-1}}, \tag{2.6}$$

де S_n – поточне значення експоненційно зваженого із ковзним середнім;

x_n – нове спостереження;

$\alpha \in (0, 1)$ – коефіцієнт згладжування, що визначає баланс між реактивністю і стабільністю оцінки.

При α близькому до 1 система майже миттєво реагувала на зміни, але оцінка була нестабільною і схильна до шумових викидів. При α близькому до 0 оцінка була стабільною, але повільно адаптувалась до тривалих змін умов. Типові значення для моніторингу лежать у діапазоні 0.1–0.3.

Інтервальна агрегація на k-му часовому проміжку обчислювалась як середнє по спостереженнях, що потрапили у проміжок, було подане формулою:

$$A\_k = x\_1 + x\_2 + \ldots + x\_(n\_k) \,/\, n\_k, \tag{2.7}$$

де A_k – агреговане значення на k-му інтервалі;

x_i – і-те спостереження у межах інтервалу;

n_k – кількість спостережень в інтервалі k.

Перевагою експоненційно зваженого із ковзним середнім є константна вимога до пам'яті O(1) на потік і обчислювальна складність O(1) на оновлення.

Метод високодинамічних гістограм для широких вибірок широко застосований у вимірювання затримок, зберігає кількість спостережень у логарифмічно розподілених діапазонах значень. При типовому діапазоні RTT від 0.1 до 10000 мс і точності 0.1% метод займає близько 80 КБ пам'яті на потік незалежно від кількості спостережень.

Жодна із перелічених стратегій не є універсальною, відповідно на практиці в стандартному вигляді вони не застосовувались, а застосовувався комбінований підхід. Із різними метриками та чи інша стратегія проявляла переваги і недоліки, але досягнути оптимальності, користуючись лише однією стратегією для всіх метрик, практично неможливо.

Вибір стратегії агрегації напряму впливає на можливості виявлення і швидкодію систем. Мережевий трафік досить складно передбачити з точки зору затримок, враховуючи, що досить поширеним явищем є короткі сплески затримок. Усереднення також вносить зміни в набір метрик для візуалізації.

Глобальні мережі не є ідеальним середовищем передачі даних, так чи інакше мають місце досить не часті аномалії, втрати пакетів, обробка із затримкою, обмеження кількості трафіку, лімітування.

2.5 Вибір методики керування системними ресурсами

Якщо розглядати методику керування системними ресурсами у контексті засобів моніторингу, то це набір способів контролювати об’єми використання часу процесора, кількість задіяної оперативної пам’яті, використання мережевого інтерфейсу, а також швидкість читання-запису даних на накопичувачі.

Споживання сервісом великої кількості системних ресурсів може бути прийнятним для повноцінних серверів, однак якщо застосовувати одноплатні комп’ютери як основну платформу, то навіть мінімальна втрата ресурсів буде суттєво впливати на оперування цієї системи.

Розглянувши існуючі open source методи, проаналізувавши їх код, було помічено, що досить часто додаткові втрати виникають на копіювання пакетів для

аналізу між просторами ядра і користувача. Стає досить очевидно, що при точкових сплесках трафіку втрати зростуть ще більше, а швидкодія сповільниться.

Важливою частиною стратегії керування ресурсами є використання ізольованих структур пам'яті для кожного ядра, що дозволяє процесору обробляти дані паралельно без необхідності використання блокувань, які в багатопотокових системах стають головною точкою обмеження і причиною непередбачуваних затримок.

Розглянувши використання оперативної пам’яті, взаємодія операційної системи відбувалася через абстракцію віртуального адресного простору, при запиті на певну кількість ресурсів фізична оперативна пам’ять не виділялась, а відбувалась резервація запису у таблицях сторінок.

Через проблеми із виділенням пам’яті на велику кількість завдань виникала проблема із фрагментацією, коли поділ на шматки ділив пам’ять для процесів не оптимально.

Обмеження на споживання ресурсів пристроєм був сформалізований системою нерівностей формулою:

$$\begin{aligned} CPU\_used &\leq CPU\_max \\ MEM\_used &\leq MEM\_max \\ IO\_used &\leq IO\_max \end{aligned} \tag{2.8}$$

де *CPU_used, MEM_used, IO_used* – поточне споживання процесорного часу, оперативної пам'яті та операцій вводу-виводу;
CPU_max, MEM_max, IO_max, – гранично допустимі значення для конкретного пристрою.

Дані переміщувались між просторами користувача і ядра, в свою чергу ця дія застосовувала процесорні цикли на перенесення а не обробку даних. Тому це є не оптимальною дією і її можна класифікувати як втрату ресурсів.

При використанні ізольованих структур пам'яті для кожного ядра загальне споживання пам'яті є сумою по ядрах, сума представлена формулою:

$$MEM\_total = MEM\_1 + MEM\_2 + \ldots + MEM\_C, \quad (2.9)$$

де *MEM_total* – загальне споживання пам'яті;
MEM_i – обсяг пам'яті, виділеної для і-го ядра;
MEM_C – кількість ядер процесора.

2.6 Логічна структура та моделювання

Побудова попередньої моделі методу є потрібним і відповідальним етапом при проєктуванні. В першу чергу, цей процес дозволяє визначити основні компоненти та зв'язки між ними, що є основою для подальшого розвитку системи. Орієнтуючись на вигляд, було оцінено загальну архітектурну гнучкість та масштабованість методу, враховуючи потреби і вимоги технічного завдання.

Основною відмінністю між застосунками прикладного рівня і системами глибокого моніторингу із точки зору попереднього планування і побудови архітектури полягали в тому, що прийняті рішення на початку напряму впливали на продуктивність, адаптації моделі – доволі трудоємкий процес. Рішення прийняті на початкових етапах одразу можуть визначити обмеження, переваги і недоліки.

Розглянувши системи, де відбувається взаємодія між системними просторами і виконавчими середовищами, узгодження і попереднє продумування роботи відбувалось ще до фактичного написання програмного коду. Будь-які помилки у визначенні меж між рівнями взаємодії, проблеми визначенні структур чи неузгодженість механізмів передачі даних призводили до ситуацій, коли стає необхідно переписувати обидві частини системи одночасно.

За своєю природою BPF і eBPF фільтри є чітко визначеними структурами, тому зміна структури подій в eBPF програмі за замовчуванням вимагала змін і Python обробника. Тому попереднє проєктування можна вважати не формальним, а більш практичним етапом, що напряму вплинув на результат розробки методу та впровадження реалізації.

Початковим кроком у попередньому моделюванні є формулювання цілей, яких необхідно було досягти, враховуючи, що це мають бути чітко визначені вимірювані вимоги. Для методів і систем пасивного моніторингу цілі було умовно поділено на функційні і нефункційні. До вимог функційних було внесено перелік того, що було необхідно перевірити і відстежувати: метрика RTT затримок при передачі даних у шляху, виявляти повторні передачі даних, відстежувати стан TCP сесій, виявляти і систематизувати активності сервісів, що активно застосовуються в конкретний момент часу. Окремо було враховано, що мав відбуватись збір всіх попередньо перелічених метрик, метрики мали проходити нормалізацію та приводитись до вигляду, щоб зовнішні системи збору могли коректно їх візуалізовувати чи доопрацьовувати.

Щодо нефункційних вимог, то сюди враховано ті чинники, які напряму визначали якісні частини системи. Метод і його реалізація не повинні були впливати на сам мережевий трафік ні в позитивну сторону, ні в негативну. Кількість ресурсів, що використовувалась підлягала окремій оцінці у вигляді окремої метрики, тобто споживання ресурсів обмежувалось конкретним значенням, що варіювалось в залежності від характеристик пристрою, на якому виконувався метод. Швидкість обробки зібраних метрик - це також не функційна вимога, тобто затримка між виявленням конкретної події і її відображенням мала бути не більше однієї секунди, це спрямовано на швидке повідомлення користувача або адміністратора мережі. Ще однією не функційною вимогою є можливість застосування методу і можливості адаптації в існуючі архітектури мереж, без додаткових модифікацій.

Оцінивши вище перелічені функційні і нефункційні вимоги, було зроблено проміжний висновок: всі вони накладали ті чи інші обмеження і робили систему більш вузькоспеціалізованою під час розробки.

Важливим етапом попереднього моделювання було управління складністю і вирішення ряду технічних завдань і проблем, не оперуючи суцільним методом, а навпаки, розділяючи метод на частини.

Під час будь-якого проєктування одним із важливих етапів є декомпозиція методу на логічно незалежні підсистеми, що мають спільні інтерфейси взаємодії. Однією із ключових ознак вдалої декомпозиції є можливість внесення змін в одну із підсистем без необхідності модифіковувати інші частини. Тобто, окрім декомпозиції, потрібна інкапсуляція підсистем, створення і визначення інтерфейсів взаємодії.

Повторно оцінивши критерії, функційні і нефункційні вимоги було проведено моделювання, визначаючи підсистеми. Першою підсистемою стала підсистема захоплення пакетів і первинної обробки подій. Ця підсистема повинна бути максимально лаконічною, здійснюючи аналіз. Результат, який дана підсистема повертає - це структуровані події, які доступні до передачі у простір користувача.

Лаконічність підсистеми обґрунтована тим, що кожна додаткова операція, визначені і невикористані функції, незастосовані змінні збільшують час обробки і задіюють більше системних ресурсів, які фактично можна було б зберегти.

Іншою підсистемою вважається обробник подій у користувацькому рівні. Переважна більшість системних ресурсів використані тут. Дана підсистема має обробляти не пакети, а безпосередньо події, що були зібрані і передані через інтерфейс попередньо.

Варто зазначити, що цілком очікувана робота на пристроях із невеликою кількістю ресурсів. Відповідно технічно створюється можливість розділити підсистеми збору і аналізу. Тобто система збору працює на пристрої із обмеженими ресурсами, в свою чергу частину аналізу винесено на віддалений сервер моніторингу для подальшої обробки. Такий підхід вивільняє частину ресурсів. Проте коли ресурсів достатньо, метод також може працювати і на єдиному фізичному пристрої.

Окремо можна виділити підсистему публікації метрик, її інтерфейс працює як HTTP сервіс і надає метрики у машинному структурованому форматі, приміром у форматі JSON або YAML. Це важливо для випадків, коли застосовуються унікальні рішення моніторингу для бізнесу чи персонального використання у

клієнтів, тому очевидним є бажання оптимізації окремих систем моніторингу і агрегування їх результатів у єдину систему.

Попереднє моделювання в даному випадку зводиться до етапів, які потрібно виконати під час безпосереднього проєктування:

1. Реалізація eBPF програми перехоплення потоків трафіку.
2. Розробка верифікатора прочитаних заголовків.
3. Додавання частини створеного методу для аналізу метрик. Метрики і особливості роботи розробленого методу будуть детально розглянуті у наступному розділі.
4. Розробка інтерфейсу передачі зібраних подій у користувацький простір.
5. Реалізація компонентів простору користувача із окремими модулями, що мають бути ізольовані один від одного.
6. Реалізація модуля, що відповідатиме за вивід результатів в термінал засобами стандартного вводу та виводу, інша частина – Prometheus модуль для уніфікації формату метрик.
7. Інтеграція і тестування в умовах реального навантаження. Поведінка роботи системи в умовах тривалої роботи.
8. Аналіз споживання системних ресурсів, пошук і виправлення можливих втрат пам’яті чи неоптимального використання ресурсів.

Попереднє моделювання і логічна структура зводиться до чіткого плану і переліку завдань, який спрямований на те, щоб розподіляти ресурси при розробці реалізації та доопрацюванні методу більш оптимально.

2.7 Висновки

У розділі 2 було проведене оцінювання можливих технічних рішень необхідних для розробки методу і реалізації моніторингу параметрів мережі. Також було здійснено попереднє проєктування архітектури. Попереднє моделювання

дозволяє оптимізувати структуру системи, щоб оптимізувати її в порівнянні існуючих рішень.

Проєктування на ранньому етапі дає змогу ідентифікувати переважну більшість технічних проблем, які потрібно розв’язати. Якість технічних рішень напряму впливала на точність, швидкодію, кількість системних ресурсів.

Для того, щоб обрати оптимальну базу, було розглянуто можливі опції вирішення тих чи інших технічних завдань.

Адаптація методу і оновлення системи на більш пізніх етапах ускладнює логіку роботи і спричиняє втрати як фінансові, так і з точки зору оптимізації.

Всі існуючі рішення є досить вузькоспеціалізованими, відповідно не можуть вирішити абсолютно всіх проблем, із якими зіштовхується користувач або адміністратор. Тому напрямок і проєктування методів і рішень в напрямку моніторингу комп’ютерних мереж із порівняно невеликою кількістю доступних системних ресурсів є досить перспективним і актуальним.

3 МЕТОД І АЛГОРИТМ СИСТЕМИ МОНІТОРИНГУ ПАРАМЕТРІВ КОМП'ЮТЕРНОЇ МЕРЕЖІ В РЕАЛЬНОМУ ЧАСІ

3.1 Метод і алгоритм аналізу параметрів

Проаналізувавши існуючі методи, що доступні у сфері, оцінивши їх спектр застосування і функції, було виявлено напрямки, де можна впровадити оптимізації.

Так як переважна більшість рішень є із закритим програмним кодом, а алгоритми і методи, що виконуються, приховані від спільноти, то було вирішено розробити новий метод і рішення відповідно. Рішення планується адаптувати як платформу для подальших досліджень та розвитку продукту. Метод планується розвивати та поступово вносити оптимізації.

Серед доступних методів збору трафіку, переваги і недоліки яких були проаналізовані попередньо, було вирішено будувати рішення багаторівневої моделі вимірювання характеристик TCP з'єднань на основі аналізу заголовків пакетів у просторі ядра операційної системи засобами eBPF. Основним методом є захоплення і фільтрація заголовків за допомогою фільтрів eBPF. Метод є досить оптимальним для одноплатних комп'ютерів, оскільки керування ресурсами системи здійснюється із порівняно меншими втратами.

Для точності і можливості широкого спостереження за мережею було визначено наступні метрики, які необхідно визначати і аналізувати:

1. Ідентифікація потоку.
2. Вимірювання RTT (двома методами для більшої точності).
3. Виявлення повторної передачі сегментів.
4. Оцінка кількості проміжних вузлів при передачі даних від джерела до точки призначення.
5. Визначення напрямку трафіку.
6. Обчислення перцентилів і різниці затримок у ковзному вікні.
7. Діагностика першопричини аномалій.

В межах етапу проєктування було проведено формалізацію вимірювання параметрів та подання метрик у математичній формалізації.

Ідентифікація мережевого потоку відбувається за умови, що пакет пройшов через конкретний мережевий інтерфейс, побудувавши ключ на основі набору даних у полях пакету.

IP-адреси належать множині натуральних чисел N (для IPv4 – 32-бітні цілі), номери портів обмежені 16-бітним представленням, ключ потоку представлений формулою:

$$k = (src\_ip, dst\_ip, sport, dport) \in N^2 \times \{0..65535\}^2, \quad (3.1)$$

де *k* – чотирискладовий ключ потоку, що однозначно ідентифікує TCP/UDP сесію на рівні транспорту;

src_ip – IP-адреса ініціатора комунікації;

dst_ip – IP-адреса призначення;

src_p, dst_p – відповідні номери портів.

Вимірювання часу RTT було проведено двома способами. Один із способів - це орієнтування на SYN/SYN-ACK стани TCP сесії, який можна використовувати як резервний, коли мережеве обладнання не підтримує опції TCP Timestamp. При встановлені сесії відправник ініціює встановлення сесії фрагментом SYN, прапорець SYN еквівалентний 1, ACK еквівалентний 0. Фільтр eBPF має зафіксувати проходження фрагменту через інтерфейс і фіксує часову мітку ядра, що представлено формулою:

$$entry\_timestamp[k] \text{ при } T\_syn = bpf\_ktime\_get\_ns(), \quad (3.2)$$

де *entry_timestamp* – часові мітки вхідних фрагментів;

T_syn – час синхронізації;

bpf_ktime_get_ns() – стандартна функція, яка повертає монотонний час у наносекундах.

При надходженні відповіді SYN-ACK сервіс моніторингу має здійснити пошук за оберненим ключем k і здійснює розрахунок RTT як різницю часових міток

між фрагментами, один із яких був ініціацією, а інший відповіддю на ініціацію, що є представлено формулою:

$$RTT1 = T\_synack - T\_syn,\ T\_synack = bpf\_ktime\_get\_ns(), \quad (3.3)$$

де *RTT1* – оцінка затримки методом триетапного рукостискання;
T_syn – збережена раніше часова мітка відправлення SYN;
T_synack – часова мітка прийому SYN-ACK.

Альтернативним засобом виміру є аналіз TCP часових міток. Згідно із RFC 7323, якщо обидві сторони підтримують опцію TSOPT, то в TCP заголовку як додаткові поля будуть доступні TSval і TSecr, тобто лічильник відправника і значення, що відповідає за останній отриманий пакет. В реалізації фільтрування і аналізу опцій за допомогою eBPF відбувається ітеративний розбір опцій представлений формулою:

$$RTT2 = T\_ack - sent\_ns,\ \text{якщо}\ tsopt\_map[k].tsval = TSecr(), \quad (3.4)$$

де *RTT2* – оцінка затримки;
T_ack – часова мітка ядра при прийомі пакета;
sent_ns – момент надсилання пакета;
tsopt_map – асоціативна таблиця;
TSecr – значення поля TSecr у спостереженому пакеті.

Значення RTT може бути коректним за умови, представленої у вигляді формули:

$$50000 \leq RTT2 \leq 5000000000\ (\text{нс}), \quad (3.5)$$

де *RTT2* – оцінка затримки;
значення *50000* та *5000000000* підібрані експериментально.

В умові достовірності верхня і нижня межа підібрані експериментально. Нижня межа потрібна, щоб не враховувати хибні спрацювання на фрагменти, що прибули в невпорядкованому форматі. Верхня межа спрямована на те, щоб не враховувати застрілі сесії, які не продовжуються впродовж 5 секунд і, можливо, не є закритими коректно.

Основний спосіб виявлення – це пошук співпадінь порядкових номерів фрагментів, що відносяться до одного і того ж потоку із різницею в часовому проміжку визначеного порогу, представленого формулою:

$$is\_ret(p) = 1, \quad last\_seq.[k].seq = SEQ(p) \text{ I } dt > T\_ret, \tag{3.6}$$

де *is_ret* – логічна функція виявлення повторної передачі;

p – спостережений пакет;

last_seq [k] – запис про попередній зафіксований пакет у потоці з ключем k;

SEQ(p) – поле порядкового номера у TCP-заголовку пакета;

dt – різниця між поточною часовою міткою і часовою міткою збереженого попереднього запису;

T_ret – мінімальний інтервал часу, що вважається повторною передачею.

Значення T_ret визначено як 50мс. Це нижня межа часу очікування повторної передачі фрагменту. Це значення дозволяє відрізнити повторну передачу від перевпорядкованих фрагментів.

Оцінку проміжних вузлів реалізовано за допомогою дослідження і варіації значень TTL. В залежності від пристрою стандартні початкові значення відрізняються, приміром Android та Linux мають стандартне значення TTL 64, операційні системи Windows та MacOS мають початкове значення 128, також в залежності від реалізацій мережевого стеку операційних систем це значення спостерігалось іншим, загальне обчислення відбувалось за формулою:

$$H = TTL_0 - TTL(p), \tag{3.7}$$

де H – оцінка кількості проміжних маршрутизаторів;

TTL_0 – стандартне початкове значення TTL ОС;

$TTL(p)$ – значення поля TTL у заголовку IP.

Значення TTL визначається як різниця стандартного значення із тим, яке потрапило у фільтр. Значні стрибки кількості проміжних вузлів на одному і тому ж потокові трафіку можуть свідчити про глобальні зміни маршрутизації.

Існують випадки, коли проміжні пристрої можуть двічі віднімати значення TTL при передачі фрагменту далі, тому різниця в одну-дві одиниці від середнього значення вважають як похибку або балансування трафіку на проміжних вузлах.

Щодо оцінювання рівня втрачених фрагментів за інтервал часу, то за одиницю часу спостереження система фіксує кількість повторних передач і загальну кількість пакетів для пристрою, що веде комунікацію. Це співвідношення кількості повторних передач до загальної кількості передач. Рівень втрат можна подати у наступному вигляді, формулою:

$$L(dt) = N\_ret / (N\_total + N\_ret), \quad (3.8)$$

де $L(dt)$ – рівень втрат пакетів за інтервал часу dt;

N_ret – кількість виявлених повторних передач;

N_total – загальна кількість переданих сегментів.

В контексті втрат було визначено рівень порогу, після перетину якого система почала сигналізувати адміністратора.

Як комплексний метод пасивного моніторингу було запропоновано замкнену систему рівнянь 3.9-3.29. Система рівнянь описує повний процес перетворень від фіксації пакету потоку у просторі ядра до висновку функції діагностики у просторі користувача. Відповідно до процесу обробки рівняння поділене на шість блоків.

Першим етапом обробки є побудова унікального ідентифікатора мережевого потоку на основі полів IP-заголовка та транспортного заголовка. Ключ потоку дозволяє системі відрізняти один потік даних від іншого серед великої кількості

пакетів, що одночасно проходять через мережевий інтерфейс, його представлено формулою:

$$f = (src\_ip, dst\_ip, src\_p, dst\_p) \in \mathbb{N}^2 \times \{0..65535\}^2, \quad (3.9)$$

де f – прямий ключ потоку.

Крім прямого ключа, будується обернений ключ, що забезпечує зіставлення запиту та відповіді у двонапрямленій комунікації, поданий формулою:

$$f^{-1} = (src\_ip, dst\_ip, src\_p, dst\_p), \quad (3.10)$$

де f^{-1} – обернений ключ потоку.

Визначення напрямку пакета (вхідний чи вихідний) є необхідним для коректної інтерпретації метрик, формалізація напрямку пакету подане формулою:

$$L = SIOCGIFCONF(interface), \quad (3.11)$$

де L – множина IP-адреса, налаштованих на мережевих інтерфейсах пристрою; SIOCGIFCONF – системний виклик Linux для отримання конфігурації мережевих інтерфейсів.

Затримки та втрати мають різну діагностичну цінність в залежності від того, який напрямок потоку, чи йдеться про вихідний трафік локального пристрою, чи про вхідний потік від віддаленого сервера, напрямок потоку визначений формулою:

$$direction(f) = 1 \text{ якщо } src \in L; \quad 0 \text{ якщо } src \notin L, \quad (3.12)$$

де *direction* – логічна функція визначення напрямку потоку;

1 – вихідний;

0 – вхідний;

L – множина IP-адреса, налаштованих на мережевих інтерфейсах пристрою.

Напрямок визначено як вхідний потік від віддаленого сервера, сформалізований формулою:

$$remote(f) \;=\; dst \text{ якщо } direction = 1; \quad src \text{ якщо } direction = 0, \tag{3.13}$$

де *remote* – функція вибору віддаленої сторони відносно локального пристрою.

Наступний етап є першим із двох способів оцінки затримки RTT. Спосіб базується на спостереженні за процесом трьохетапного рукостискання TCP-сесії: система фіксує момент відправлення SYN-фрагмента та момент отримання SYN-ACK у відповідь, етап поданий формулою:

$$entry[f] <- bpf\_ktime\_get\_ns(), \text{ якщо } SYN, \tag{3.14}$$

де *entry* – асоціативна таблиця, у яку фіксуються часові мітки;
bpf_ktime_get_ns – системна функція, що повертає часову мітку.

Різниця цих часових міток дає оцінку RTT для конкретного потоку, визначення різниці представлене формулою:

$$RTT1(f) \;=\; bpf\_ktime\_get\_ns() \;-\; entry[f^{-1}] \text{ при } SYN + ACK, \tag{3.15}$$

де *RTT1* – оцінка затримки методом рукостискання.

Цей спосіб є резервним і застосовується у випадках, коли мережеве обладнання або кінцевий пристрій не підтримує опції TCP Timestamp. Перевагою є його простота та сумісність з будь-якими TCP-з'єднаннями, недоліком – можливість виміряти RTT лише одноразово на момент встановлення сесії.

Другий спосіб вимірювання RTT використовує опцію TCP Timestamp. На відміну від виміру рукостискання, який працює лише під час встановлення з'єднання, спосіб TCP Timestamp вимірює RTT протягом усього часу існування

сесії для кожного пакета. Система виконує ітеративний розбір TCP-опцій у просторі ядра, розбір представлено формулою:

$$(TSval, TSecr) = parse\_tspopt(skb, ETH\_HLEN + ip\_hl + 20,\ ETH\_HLEN + \\ + ip\_hl + tcp\_hl, \quad (3.16)$$

де *parse_tspopt* – функція ітеративного розбору опцій TCP-заголовка;
skb – структура буфера сокета ядра, що містить пакет;
ETH_HLEN – розмір Ethernet-заголовка;
ip_hl – розмір IP-заголовка;
tcp_hl – розмір TCP-заголовка.

Зберігання пари (TSval, часова мітка) в асоціативну таблицю відбувається у за допомогою формули:

$$tsmap[f^{-1}] <- \{TSval, T\_sent = bpf\_ktime\_get\_ns()\} \text{ якщо } TSval > 0, \quad (3.17)$$

де *tsmap* – асоціативна таблиця, що зберігає пари (TSval, час_надсилання) за оберненим ключем.

При отриманні відповіді з відповідним TSecr обчислює різницю часових міток, на основі формули:

$$RTT2(f, p) = bpf\_ktime\_get\_ns() - tsmap[f].T\_sent, \text{ якщо } TSecr(p) = \\ tsmap[f].TSval, \quad (3.18)$$

де *RTT2* – оцінка затримки методом TCP Timestamp для потоку.

Результат проходить перевірку на достовірність, значення в діапазоні від 50 мкс до 5 с вважаються коректними, що представлено формулою:

$$valid(RTT2) = 1 \text{ якщо } 5 * 10^4 \leq RTT2 \leq 5 * 10^9 \text{нс.}, \quad (3.19)$$

де *valid(RTT2)* – логічна функція перевірки результату вимірювання RTT2.

Підсумкове значення RTT обирається з пріоритетом TCP Timestamp як більш точного, що представлено формулою:

$$RTT(f,k) \;=\; RTT2(f,k) \text{ якщо } valid = 1; \quad \text{інакше } RTT1(f), \tag{3.20}$$

де *RTT(f, p)* – підсумкове значення затримки;

f – потік;

p – пакет.

Повторна передача TCP-сегмента є одним із основних індикаторів проблем на мережевому шляху: втрати пакетів, перевантаження каналу або нестабільності маршрутизації. Виявлення повторних передач базується на порівнянні порядкового номера (SEQ) поточного сегмента із раніше зафіксованим номером для цього ж потоку, представленого формулою:

$$is\_retr(f,p) \;=\; 1, \text{ якщо } SEQ(p) = last \text{ і } dt > 5 \cdot 10\text{^}7 \text{ нс.}, \tag{3.21}$$

де *is_retr (f, p)* – індикатор повторної передачі сегмента p у потоці f;

last – запис про останній зафіксований сегмент потоку;

SEQ(p) – порядковий номер у TCP-заголовку пакета p;

dt – різниця часових міток.

Якщо номери збігаються, а різниця часових міток перевищує поріг 50 мс, сегмент класифікується як повторно переданий. Поріг необхідний для відокремлення повторних передач від перевпорядкованих сегментів, які можуть прибувати з незначною затримкою через особливості маршрутизації, визначений формулою:

$$last[f] \;<-\; \{SEQ(p), bpf\_ktime\_get\_ns()\}, \tag{3.22}$$

де *last [f]* – пара із значень із порядковим номером, часовою міткою.

На цьому етапі методу було виконано два основних завдання. Метод визначає початкове TTL через вибір найближчого стандартного значення (64, 128 або 255), поданого формулою:

$$TTL_0(p) \;=\; min\{\, t \in \{64, 128, 255\} : t \geq TTL(p) \,\}, \tag{3.23}$$

де *TTL₀ (p)* – оцінка початкового значення TTL для пакета p;
min – функція вибору найменшого значення;
TTL(p) – реконструкція оригінального TTL відправника.

Перша – оцінка кількості проміжних маршрутизаторів (вузлів) на мережевому шляху за допомогою аналізу поля TTL у IP-заголовку. Кожен маршрутизатор зменшує значення TTL на одиницю, тому різниця між початковим значенням TTL операційної системи відправника та виміряним значенням дає оцінку кількості проміжних вузлів, що було подно формулою:

$$H(f) \;=\; TTL\_\text{поч}(p) \;-\; TTL(p), \tag{3.24}$$

де *H(f)* – оцінка кількості проміжних маршрутизаторів на шляху потоку f;
TTL(p) – реконструкція оригінального TTL відправника.

Для уніфікації зберігання метрик використовується формула:

$$k\_c(f) \;=\; min\{\, (src, sp, dst, dp), (dst, dp, src, sp) \,\}, \tag{3.25}$$

де *k_c (f)* – ключ потоку, застосовується для уніфікованого зберігання стану потоку незалежно від напрямку.

Друге завдання – відстеження стану TCP-сесії через скінченний автомат, який класифікує кожне з'єднання у один із станів - SYN_SENT, ESTABLISHED, FIN або RST, автомат сформалізований формулою:

$$state(k\_c) \;\in\; \{SYN\_SENT, ESTABLISHED, FIN, RST\}, \tag{3.26}$$

де *state (k_c)* – поточний стан скінченного автомата TCP-сесії.

На даному етапі відбувається агрегація зібраних значень RTT для статистичного аналізу. Ковзне вікно фіксованого розміру (N = 500) зберігає останні вимірювання затримки для кожного потоку і реалізоване як кільцевий буфер, представлений формулою:

$$W(f) \;=\; \{\, RTT(f,n), RTT(f,n-1), \ldots, RTT(f,n-N+1) \,\},\;\; N \;=\; 500, \qquad (3.27)$$

де *W(f)* – ковзне вікно останніх N значень RTT для потоку f;
n – номер поточного спостереження;
N = 500 – розмір вікна.

На основі вмісту вікна обчислюються три значення за допомогою формули:

$$P\_p(W) \;=\; W\_sorted[\, round\_up(p/100 \cdot W) \;-\; 1 \,],\;\; p \;\in\; \{50, 95, 99\}, \qquad (3.28)$$

де *P_p (W)* – перцентиль рівня p від вибірки W;
W_ sorted – вибірка W, відсортована за неспадною затримкою;
W – потужність вибірки (кількість елементів);
round_up – операція округлення до більшого цілого;
p – значення перцентилів.

Перцентилі визначають типову та граничну поведінку затримок: медіана визначає типове значення, а перцентилі визначають аномальне відхилення, що часто свідчать про перевантаження мережі або аномалії. Різниця затримок обчислюється як середнє абсолютних різниць між сусідніми значеннями RTT і характеризує стабільність каналу, різниця затримок, висока затримка типова для перевантажених або нестабільних мережевих шляхів, низька – для стабільних з'єднань із передбачуваною поведінкою, обчислення представлене формулою:

$$J(W) \;=\; (\, \Sigma i \; RTT(f,i) \;-\; RTT(f,i-1) \,) \,/\, (W \;-\; 1), \qquad (3.29)$$

де *J(W)* – різниця затримок по вибірці W, обчислюється як середнє абсолютних різниць сусідніх значень RTT;

W – потужність вибірки (кількість елементів).

Окремої уваги заслуговує вирішення проблеми із віддаленим діагностуванням завантаженого мережевого буферу пристроїв, що здійснюють отримання даних.

У протоколі TCP кожен сегмент несе дані у полі window, це шістнадцяти бітне значення, яке напряму вказує відправнику, який об'єм даних може прийняти отримувач. Значення sk_rcvbuf у ядрі операційної системи отримувача впливає на те, що буде вкладено в мережеві сегменти і передано відправнику. Механізм керування потоком в цілому є загальноприйнятим і визначеним RFC, але досить часто для аналізу з'єднань ним часто нехтують. Звичайний користувач може спостерігати проблему із зниженням швидкості завантаження, вважати, що проблема знаходиться десь у мережі, а фактично пристрій просто не може опрацювати той потік, який йому надсилається.

Заповнений мережевий буфер пристрою напряму негативно впливає на швидкість завантаження і для інженерів, що не мають глибокого досвіду роботи із мережами, це може бути не очевидним фактором. Навіть зміна надавача послуг доступу до глобальної мережі може не повпливати позитивним чином на занижену швидкість завантаження файлів конкретним пристроєм. Проблема "порожнього вікна" передачі даних створює ситуацію, коли завантаження зупиняється практично повністю, без очевидних, із першого погляду для користувача, причин.

Зазвичай діагностику проблеми можливо провести на безпосередньо пристрої, що завантажує дані - це очевидний недолік. З'являється необхідність виконувати ряд команд, аналізувати вивід в ручному режимі, переривати діяльність користувача, якщо мова йде про корпоративну мережу. Як класичний метод діагностики можливо перевірити /proc/net/sockstat для подальшого аналізу мережевих буферів. Цей підхід не є зручним та очевидним.

Тому метод пропонує інше вирішення цієї проблеми, аналіз сегментів в потоці на проміжному вузлі, фіксація розміру вікна у сегментах TCP, що

передаються у момент аналізу. Ця частина методу може бути побудована поряд із рештою у просторі ядра на основі eBPF. Фіксація і передача значення у простір користувача дозволяє здійснити аналіз без втручання в роботу користувача пристрою, що отримує потік даних та в цілому без ручної перевірки значень, що усуває можливість людської помилки.

Коли сесія встановлена і починається отримання даних, розмір вікна передачі даних росте експоненційно до моменту, поки не відбувається втрата сегменту, механізм роботи TCP визначений стандартами RFC. За замовчуванням, максимальним значенням вікна передачі даних є 65535 біт, тобто 65КБ. Тобто від моменту встановлення сесії розмір за нормальних умов може зростати від 0 до 65535 і це стандартна робота, яку не можна вважати аномалією.

Однак метод передбачає можливу проблему, коли значення зростало, а потім різко знизилось до 2000 байт, при цьому всі сегменти потоку рухались в напрямку пристрою призначення, тобто отримувача, а в цей час пристрій-отримувач даних вирішив знизити розмір вікна передачі даних із власної ініціативи. Так можна виявити аномалію, відмінну від класичної втрати пакетів на шляху і повідомити користувача пристрою чи адміністратора про можливо проблему конкретного пристрою, яку потрібно продіагностувати і вирішити.

У випадку виявлення аномалії стан класифікується як “PC-OVRLD” і вказує на можливу проблему. Метод виявляє проблему на проміжних вузлах, що мають фізичне з’єднання у напрямку до пристрою, для якого потрібно здійснити аналіз без втручання в роботу системи в момент, коли спостерігається проблема.

Власне завдяки аналізу заголовка сегментів можна відрізнити стан “PC-OVRLD” від класичних мережевих втрат. Існуючі мережеві аналізатори та рішення, на основі існуючих методів, не дозволяють виявити проблему такого типу без застосування агентів на кінцевому пристрої або засобів глибокого аналізу пакетів, що розташовані у ядрі мережі.

3.2 Особливості впровадження методу

Розглянувши попередньо опції, які були використані для програмної реалізації, оцінивши ризики, було прийнято рішення застосовувати комбіноване рішення із мов програмування C та Python.

Мова програмування C є необхідною для взаємодії із ядром і фільтром eBPF, це забезпечує низькорівневу обробку трафіку, без переміщень пакетів у користувацький простір. Швидкодія мови програмування C є перевагою для загальної швидкодії системи.

Мова програмування Python в контексті побудови програмної реалізації вирішує інше завдання. Створення програмних інтерфейсів для міжсервісної взаємодії, обробки і аналізу тих результатів, які були отримані від eBPF, основна функція, на яку планується використати мову програмування Python. Акцент на швидкодії робиться у тому місці, де можливе надвисоке навантаження через сплески трафіку.

Технологія eBPF є не зовсім ідентичною до класичних підходів програмування у користувацькому просторі. Так як виконання відбувається у просторі ядра, відповідно синтаксис і доступні конструкції є обмеженими. Обмеження пов’язанні із роботою верифікатора ядра перед безпосереднім завантаженням коду.

Статичний верифікатор ядра перевіряє кожну процедуру і конструкцію до виконання. Верифікатор будує граф потоку управління і симулює виконання всіх шляхів, аналізуючи зміни регістрів і поведінку програми. Будь - яке відхилення від дозволених конструкцій спричиняло виведення діагностичного повідомлення і виконання було неможливим до ре-конфігурації синтаксису до норми. Навіть за умови успішної компіляції код може бути недоступним до фактичного виконання.

При розробці алгоритмів потрібно окремо від вимог компілятора акцентувати увагу і на вимоги верифікатора. Суттєвим і помітним обмеженням є заборона на використання циклів без статичної термінації і чітко визначеної кількості ітерацій

виконання. Приміром цикл із break не може бути верифікованим, навіть якщо гарантовано впевнені в існуванні умови, при якій цикл закінчиться.

При створенні eBPF програм потрібно передбачити механізм передачі станів. BPF мапа - це структура, що існує в просторі ядра, але водночас доступна із простору користувача. Для хеш-таблиць BPF_HASH доступні декілька операцій - пошук, оновлення і видалення. Особливістю у порівнянні із хеш-таблицями є те, що BPF_HASH таблиці мають статичний розмір, під який виділяється пам’ять у момент запуску. Перелічену специфіку не можна позиціонувати як недолік, це правила, яких було необхідно дотримуватись і враховувати при написанні коду.

У методі, надлишкові дані між просторами ядра і користувача не переносяться, щоб зберегти ресурси системи, однак повного перенесення уникнути не вдасться. Технічно передача даних між просторами можлива завдяки окремим вказівникам запису на рівні ядра і читання на рівні користувача.

Тобто, якщо потрібно перенести значимі поля пакету розміру 40-80 байт замість повних 1500 байт, то на рівні ядра записуються дані в буфер, виконуючи функцію perf_submit, а для зчитування в просторі користувача виконується функція perf_buffer_pool.

Через надмірну кількість трафіку і високу інтенсивність, яку операційна система не могла опрацювати через брак ресурсів, відбувалося переповнення буферу, яке могло здійснити вплив на загальну статистику.

Для точного обчислення інтервалів часу, оптимальним варіантом є використання часу ядра у наносекундах, замість сторонніх залежностей. Опитати час ядра можна завдяки функції bpf_ktime_get_ns, із точність до 1нс.

Алгоритм блок-схеми частини рішення, що виконується у просторі ядра зображено на рис. 3.1.

Поділ виконання частин методу на два окремих простори також спрямований на підвищення безпеки системи. Взаємодія між просторами відбувається у формі однонаправленого потоку. З міркувань практичності і безпеки поділ реалізований так, щоб у просторі ядра відбувається доступ і фільтрація потоків трафіку, а у просторі користувача виконувались частини, для яких необхідна динамічна

пам’ять. В користувацькому просторі опрацьовувались складні структури даних, оскільки саме там доступна підтримка більшої кількості бібліотек мов програмування. Також у користувацькому просторі було більш доцільно проводити агрегацію статистик.

Як інструмент взаємодії між мовами програмування Python і C застосована бібліотека BCC, під час виконання частини Python коду, відбувалась компіляція C коду. Такий підхід дозволяв вносити зміни в C код і тестувати результати одразу на пристрої, без необхідності використання бінарних файлів. Для застосування бібліотеки необхідно було передавати метадані ядра.

Відлагодження eBPF застосунків є значно складнішим порівняно із звичайним C кодом, для тестування і фіксації виведень застосовується трасування ядра за шляхом /sys/kernel/debug/tracing/trace_pipe. Тобто для проведення тестувань скриптів не було можливості виведення результатів відлагоджування в термінал через засоби стандартного виводу, результати можна було побачити після виконання, переглянувши вміст файлу.

Побудова методу та системи зведена до первинної фільтрації і агрегації метрик у просторі ядра і доопрацювання в просторі користувача. Даний підхід і вирішує проблему управління ресурсами і забезпечує стабільну роботу на пристроях із невеликою кількістю ресурсів, в тому числі і одноплатних комп’ютерів.

У порівнянні із існуючими системами побудованими на основі методів, доступними на ринку, цей підхід демонструє приріст у продуктивності та зменшення використання системних ресурсів через вдосконалений засіб обробки потоків трафіку.

Протягом всього часу виконання, відбувалось вичитування записів із буферу ядра через інтервал у 100мс. Функція виконується у однопоточному режимі, механізм псевдо багатопоточності у скрипті застосовується для DNS запитів та сервера метрик Prometheus.

Як було зазначено раніше, на етапі виконання вивчались локальні IP адреси на основі активних мережевих інтерфейсів, щоб спростити процес ідентифікації

пакетів на вхідні і вихідні. У цьому просторі відбувалось безпосереднє обчислення і визначення параметрів на основі отриманих даних у необробленому вигляді.

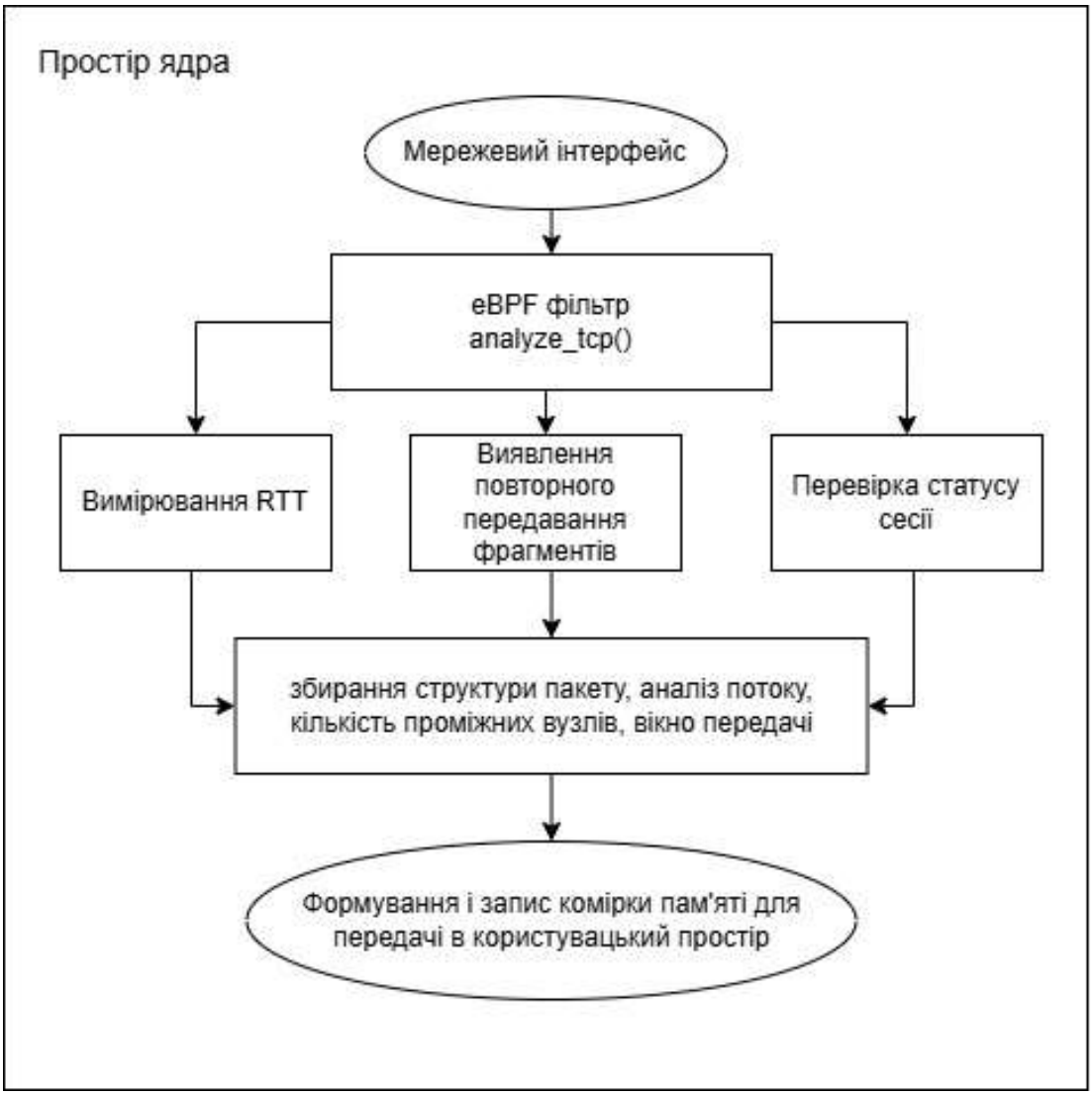


Рисунок 3.1 – Вигляд роботи методу в частині, що виконується у просторі ядра

Як додаткова функція, на основі IP адреси отримувача відбувалось виконання функцій, спрямованих на визначення мережевої автономної системи (ASN). Адміністратор може використати номер автономної системи при пошуку точок, де спостерігаються втрати та при розслідуванні мережевих аномалій. Дана функція не блокує головний процес від продовження виконання так як DNS запити можуть виконуватись декілька сотень мілісекунд. Блокування головного процесу на час

очікування відповіді від DNS сервера може спричинити затримки через зупинку перехоплення потоків.

У просторі користувача також відбувається етап діагностики і класифікації аномалій. Пошук аномалій відбувається на основі чотирьох правил, які перевіряють всі потоки на основі зібраних діагностичних деталей. Правила перевіряються незалежно, на основі умов відбувається формування результату. Це спрямовано на те, щоб адміністратору вказати на можливі аномалії. Передбачено чотири сценарії, окрім штатної роботи мережі: втрати пакетів під час передачі із підвищеними затримками, втрати пакетів без підвищених затримок, нестабільність шляху або перевантаження буферу, також можлива ситуація, коли точка призначення знаходиться на великій географічній відстані, відповідно значення RTT може коливатись, але без втрат пакетів.

Керуючись існуючими функціями та особливостями методу, була побудована діаграма частини функцій у користувацькому просторі, рис. 3.2. Діаграма є спрощена для розуміння загальної структури, без особливостей, що стосуються реалізації. Одним із кроків для вдосконалення методу був пошук рішень для проблем розпаралелювання завдань і кластеризації.

Результати попереднього моделювання архітектури є достатньо корисними на цьому етапі, так як можливо рухатись починаючи від реалізації у просторі ядра до прикладного рівня, уникаючи ускладнень в побудові взаємодії. Це допомогло розрахувати можливу кількість ресурсів та сприяло спрощенню логіки на користь кращої швидкодії. Оптимальне попереднє планування із переліком ключових етапів дійсно допомогло виявити і здійснити оптимізацію схеми, за якою працює метод. Визначені ще на етапі планування інтерфейси взаємодії між рівнями також при побудові практичної реалізації стали корисним кроком, що відобразився значним чином саме на можливостях уніфікації та масштабування.

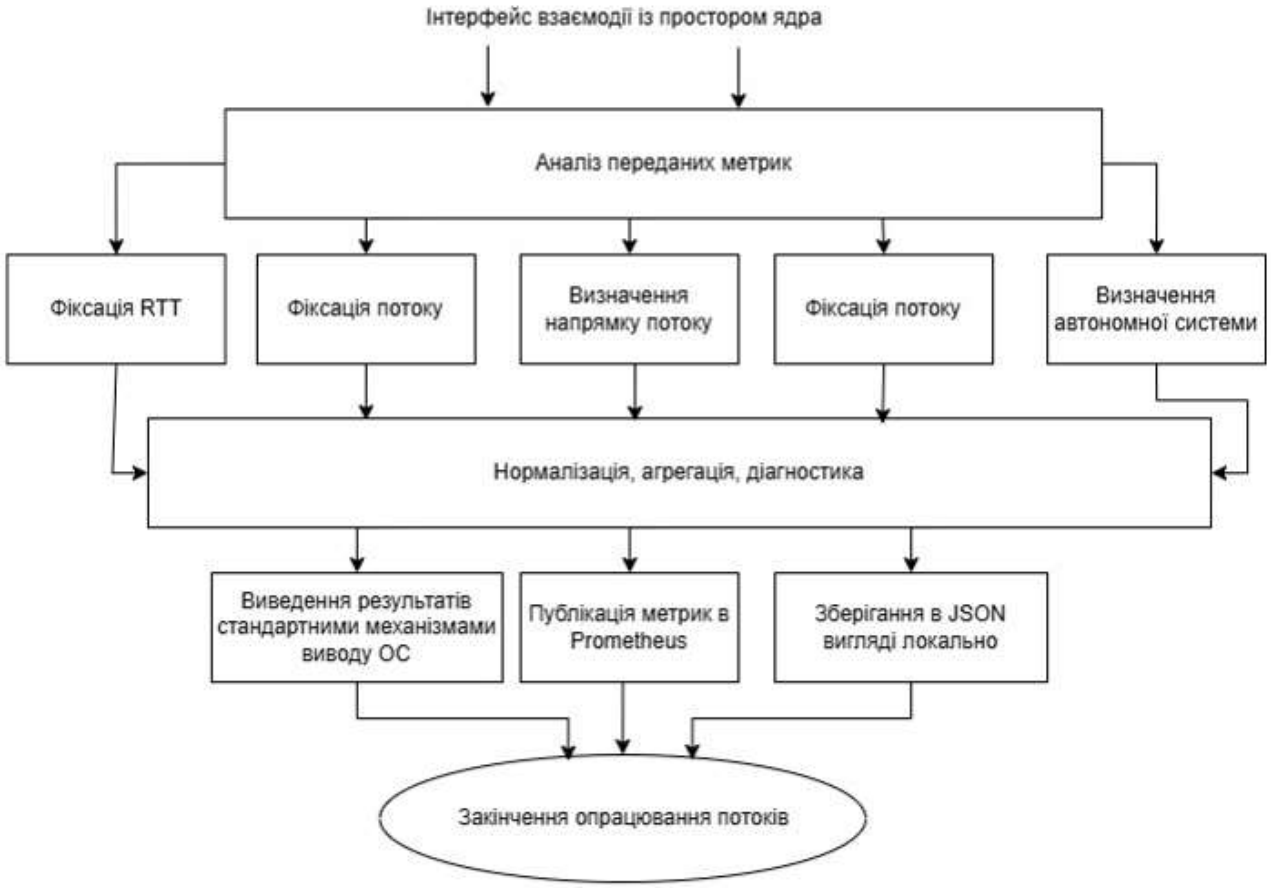


Рисунок 3.2 – Вигляд методу в частині, що виконується у просторі користувача

3.3 Порівняльний аналіз розробленого методу із існуючими методами і рішеннями

Запропонований метод відрізняється суттєвими особливостями роботи порівняно із існуючими методами і засобами. Для того, щоб порівняти наш метод із існуючими рішеннями, було проведено порівняльний аналіз на основі набору критеріїв, результати подані у таблиці 3.1.

Серед значимих особливостей можна вирізнити двоканальне вимірювання затримок із автоматичним переключенням способу відповідно до стану, агрегацію у просторі ядра з передачею у користувацький простір подій замість пакетів та інтегровану діагностику першопричин аномалій.

Таблиця 3.1 – Порівняльний аналіз розробленого методу із існуючими рішеннями, побудованих на основі альтернативних методів

Критерій	Ping/ MTR	Tcpdump	NetFlow	Datadog NPM	ntopNG	Розроблений метод
Затримка на потік (RTT per flow)	ICMP/ UDP RTT	-	-	+	+	+ (двома способами)
Затримка на пакет (RTT per packet)	-	-	-	+	-	+
Повторна передача	-	+ (вручну)	-	+	-	+
P50/p95/p99 RTT	-	-	-	-	+	+
Робота без хмарної інфраструктури	+	+	+	-	-	+
Необхідна кількість ресурсів	Низька	Середня	Середня	Висока	Висока	Середня

Порівнявши запропонований метод, можна отримати результат, на основі якого помітно, що метод вирішує проблему аналізу різних метрик, у вигляді єдиного засобу. Це технічне завдання, яке не є повністю розв’язаним за допомогою існуючих методів і засобів у ніші систем аналізу мережевого з’єднання. Точність

виміру з середньою кількістю використання ресурсів все ще досяжна для запуску реалізації методу на однопалатних комп’ютерах під керуванням операційної системи Linux. Відсутність залежності від зовнішньої інфраструктури та легка інтеграція у існуючу мережу - це безпосередньо переваги реалізації за рахунок того, що архітектура є більш продуманою.

До значимих переваг також внесено двоканальне вимірювання RTT з формальним критерієм переключення методів, агрегація метрик у просторі ядра з фіксацією часових міток, окремо було впроваджено вбудований засіб діагностики, що розрізняє проблеми мережі і локальні проблеми. Перелічені переваги не є доступними у жодному із розглянутих аналогів без розгортання додаткових рішень і додаткової інфраструктури

3.4 Висновки

У розділі 3 відбулось власне розробка методу, який спрямований на те, щоб вирішити ряд технічних завдань, що стосуються моніторингу параметрів мережі із мінімальної кількістю застосованих ресурсів, адаптований для використання на одноплатних комп’ютерах із швидкістю виявлення аномалій близькою до реального часу.

Метод будувався так, щоб створити новий засіб, який вирішує ті проблеми моніторингу, які на даний момент не можуть повноцінно вирішити існуючі засоби доступні на ринку, у вигляді єдиного технічного рішення.

Всі підходи та технічні рішення, що застосовуються, були обґрунтовані з врахуванням технічного завдання та визначених напрямків оптимізації.

При створенні реалізації на основі розробленого методу окрема увага приділялась етапам проєктування архітектури, це було зроблено для того, щоб створити уніфіковане рішення, із високими можливостями інтеграції у існуючі інфраструктури. Реалізація передбачає взаємодію модулів у різних системних просторах, з метою збереження системних ресурсів.

Для наступного етапу було описано, створено і обґрунтовано ті метрики, які є критичними для етапу тестувань. Окремо було проаналізовано існуючі методи, порівнюючи їх можливості із нашим методом на основі конкретних практичних вимірів і застосувань. Як було зазначено у таблиці 3.1., наш метод демонструє більш широкий спектр можливостей для аналізу мережевих параметрів та ідентифікації аномалій.

Практична реалізація, що побудована на основі нашого методу, містить ряд кроків, спрямованих на уніфікацію та можливості інтеграції її у існуючі системи моніторингу бізнесу або користувача. Використання ресурсів є порівняно невеликим із існуючими системами, відповідно це дає можливість запуску системи на одноплатному комп’ютері, відповідно зменшує вимоги до бюджету, який необхідно витратити бізнесу для розгортання подібних систем, що є окремою перевагою над доступними рішеннями.

4 РЕЗУЛЬТАТИ РОБОТИ МЕТОДУ І РЕАЛІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ ПАРАМЕТРІВ КОМП'ЮТЕРНОЇ МЕРЕЖІ В РЕАЛЬНОМУ ЧАСІ. ЕКСПЕРЕМЕНТИ ЗДІЙСНЕНІ ВІДНОСНО СИСТЕМИ

4.1 Виконання виміру використання ресурсів при роботі засобу аналізу мережевих з'єднань, порівняння використання ресурсів із існуючими засобами

Оцінка ресурсоємкості засобу мережевих з'єднань є окремими завданням і окремою проблемою, що вимагає вирішення. Етап оцінки - це одна із частин верифікації роботи методу і його реалізації.

Середовище проведення тестування було наступне: метод і реалізація виконуються на однопалатному комп'ютері Raspberry Pi 4B під керуванням операційної системи Raspbian 11.

Так як реалізація методу працює не лише в користувацькому просторі, а і у просторі ядра, то і процес виміру також є не зовсім стандартним процесом. Для підтвердження ефективності потрібно здійснити вимір процесорного часу, оперативної пам'яті, задіяної пам'яті для побудови eBPF мап та загальну затримку обробки. Всі перелічені вироби вимагають окремого набору утиліт, інструментів і методів.

Виміри розпочато із навантаження на процесор та кількість задіяного процесорного часу. Оскільки активна обробка відбувається у двох просторах: виконання eBPF фільтрації у просторі ядра і виконання Python скриптів для подальшого аналізу у просторі користувача, то для виміру необхідно застосовувати різні відповідні способи.

Частину у просторі ядра було виміряно інструментом bpftool prog show, як основний вивід отримано статистику виконання викликів та загальний час виконання у наносекундах, приклад виводу продемонстровано на рис. 4.1.

Рисунок 4.1 – Вимір часу виконання

На рисунку було представлено результати двох послідовних викликів утиліти, що відображають статистику виконання реалізації запропонованого методу у просторі ядра операційної системи Linux. Між двома вимірюваннями було виконано затримку 60 секунд, щоб провести оцінку динаміки роботи під навантаженням. Із ключових значень можна виділити обсяг заблокованої пам’яті ядра, що не підлягає витісненню у файли підкачки у розмірі 24576 байт, значення xlated 23568 байт - розмір транслованого eBPF коду після проходження верифікатора ядра, run_time_ns – час виконання.

Повторний виклик bpftool з фільтрацією через grep показує зміну лічильників run_time_ns та run_cnt. Різниця між двома вимірюваннями демонструє середній час виконання методу на один пакет протягом інтервалу спостереження.

П’ять BPF-мап забезпечили зберігання стану потоків, часових міток TSval, записів про останні порядкові номери сегментів, стану TCP-сесій та лічильників метрик. Розмір JIT-скомпільованого коду підтвердив компактність реалізації, що є демонстрацією можливостей виконання в обмеженому середовищі верифікатора eBPF.

Стабільність середнього часу виконання між вимірюваннями підтвердила відсутність зниження продуктивності при тривалій роботі та відповідність встановленим оптимальним значенням.

Для вимірювання кількості ресурсів, яка споживається Python процесом було застосовано утиліту perf stat, для PID процесу. Для порівняльного аналізу

використання процесорного часу потрібно провести виміри при різних значеннях навантаження: низьке - 10 Мбіт/с, середнє - 50 Мбіт/с, високе - 100 Мбіт/с.

Вимірювання оперативної пам’яті можливо провести аналізуючи /proc/[PID]. Із значимих полів виводу найбільш важливим для аналізу є VmRSS – використання фізичної пам’яті, VmPeek – максимальне значення використання оперативної пам’яті за час роботи та VmSwap – кількість задіяної пам’яті із файлів підкачки.

Для проведення виміру пам’яті задіяної для побудови eBPF мап не достатньо перевірити значення у /proc/[ID]/status, так як це адресний простір ядра. Тому для вимірів потрібно знову звернутись до bpftool із аргументами map show. Як вивід отримано ідентифікатор, тип, розмір ключа, максимальну кількість записів і поточне заповнення BPF мапи. Щоб більш динамічно відстежувати зміну значень протягом тесту можна виконати набір команд “watch -n 1 ‘sudo bpftool map show’”. Загальна місткість записів одноплатного комп’ютера становить приблизно 10240 записів.

Вимірювання загальної затримки сервісу визначає затримку між передачою даних із простору ядра у простір користувача. Для виміру найпростішим способом є додавання функції у існуючий Python скрипт. Оптимальним результатом є якнайменше значення затримки. Коректною роботою можна вважати значення затримки до 100-200 мс, цей поріг значень не впливає негативно на результати аналізу стану мережі на основі метрик.

Сформуємо результати, які пропонуються як оптимальні: середній час виконання eBPF функції на один пакет не повинен перевищувати 10 мкс., загальна кількість використання ресурсів ядра при навантаженні 100Мб/с немає перевищувати 50% ядра. Споживання оперативної пам’яті при 1000 активних потоках очікується у діапазоні 150-500 МБ. Заповнення BPF мап немає перевищувати значення у 80% від загальної кількості буфера. Кількість втрачених подій має бути нулевим значенням, оскільки воно напряму впливає на коректність роботи методу і реалізації.

Перевищення будь-якого із значень вимагало коригуючих дій або означало, що пристрій, на якому виконується сервіс, не містить необхідної кількості ресурсів.

На рис. 4.2 зображено результати після виміру, фіксації і підрахунку споживання системних ресурсів при виконанні запропонованого методу.

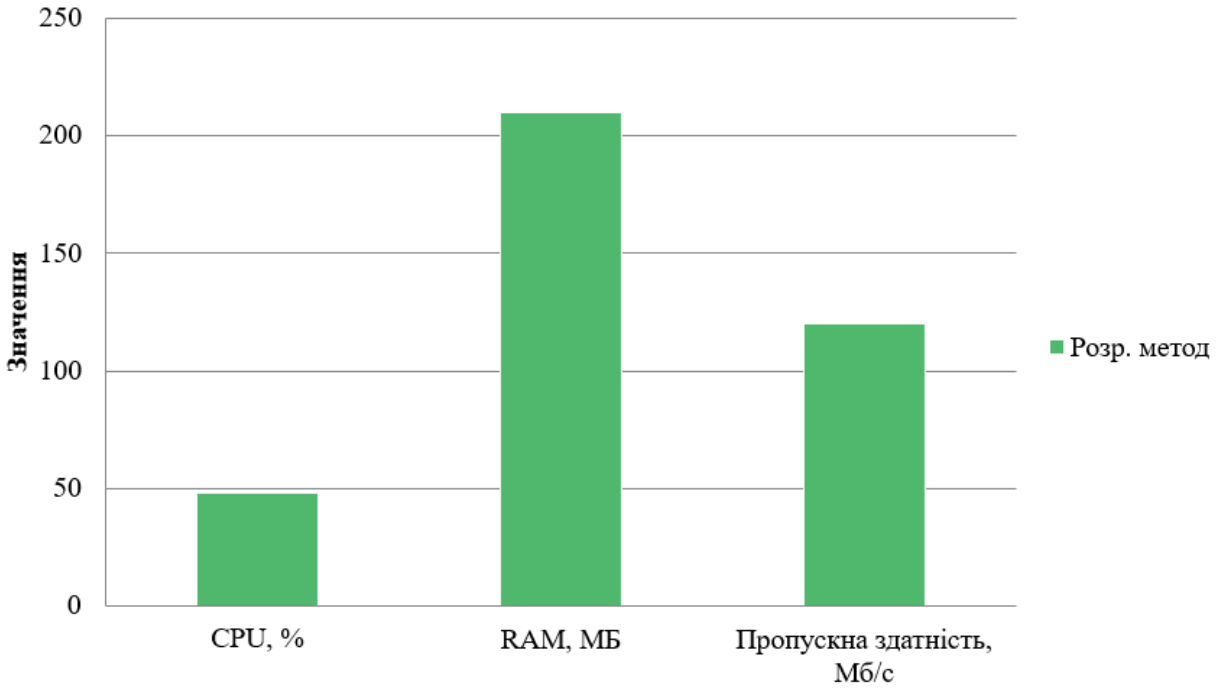


Рисунок 4.2 – Результати отримані після виміру споживання системних ресурсів при виконанні запропонованого методу

Було виконано вимірювання споживання ресурсів, коли не відбувається активного використання мережевих інтерфейсів і прийом-передача трафіку є мінімальними. В цей момент часу кількість використаних ресурсів процесора становить 0.6%, 0.0%, 3.2% та 0.0% для кожного із ядер відповідно, загальна кількість використаної оперативної пам’яті є близька до 426МБ, рис. 4.3.

У ході дослідження було проаналізовано можливість зниження навантаження на систему шляхом відключення підсистеми експорту метрик Prometheus та модуля запису подій у файл. Замість цього було використано режим виведення результатів виключно у термінал, що дозволило суттєво зменшити споживання ресурсів.

Після відключення Prometheus було звільнено один мережевий порт та усунуто періодичні HTTP-запити. Було ліквідовано витрати на серіалізацію метрик у текстовий формат, що потребувало обходу всіх внутрішніх структур даних та форматування рядків.

Споживання оперативної пам’яті знизилось на 30-50МБ, навантаження на процесор знизилось на 3-7%.

Проведений експеримент показав, що для систем із обмеженими ресурсами, де не потребується довготривале зберігання метрик або інтеграція із зовнішніми системами візуалізації, метод може ще більш оптимально використовувати системні ресурси.

Рисунок 4.3 – Споживання ресурсів системою, коли трафіку у мережі мінімальний

Для порівняння запустимо завантаження файла значного об’єму та проведемо заміри знову. Як можна побачити на рис. 4.4, при наростанні кількості мережевого трафіку збільшує кількість системних ресурсів необхідних для опрацювання цього трафіку. Завантаження процесора збільшилось до 2.2%, 43.4%, 32.9% та 16.4% для кожного із ядер відповідно. Використання оперативної пам’яті зросло до 450МБ.

Цілком очікувано, що кількість трафіку напряму впливає на системні ресурси, що необхідні для його опрацювання та аналізу. Однак рішення побудоване

на основі одноплатного комп’ютера цілком достатньо для проведення обчислень та аналізу такого масштабу.

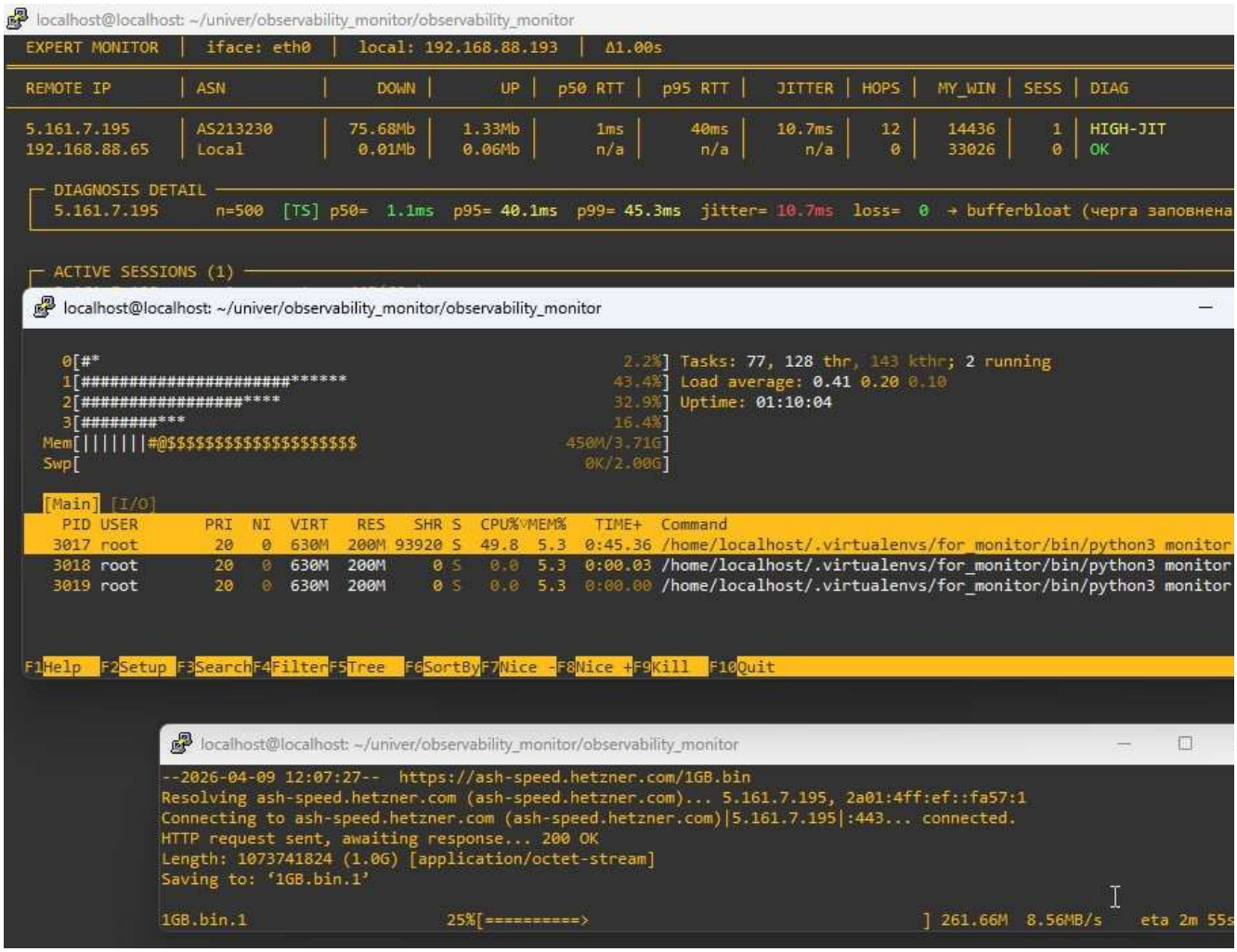


Рисунок 4.4 – Зростання використання ресурсів відповідно до збільшення мережевого трафіку

Для порівняння здійснено вимір звичайного зеркалювання трафіку з використанням утиліти tcpdump за конкретним портом, для тестування було обрано 443 TCP порт і запущено сканування, записуючи результати у файл, при цьому зафіксовано споживання ресурсів, процес зображено на рис. 4.5.

Результати продемонстурвали, що зеркалювання без жодного аналізу використовувало 25.5%, 21.3%, 4.6%, 1.3% ресурсів ядер процесора відповідно та 353МБ оперативної пам’яті.

Кількість спожитих ресурсів є частково меншою в порівнянні із ресурсами необхідними для нашого методу, однак варто враховувати те, що tcpdump лише

фіксує пакети. Жодного аналізу в автоматичному режимі не відбувається, будь-який детальний аналіз необхідно проводити в ручному режимі. Тобто утиліта tcpdump при практично еквівалентному використанні системних ресурсів не може забезпечити аналіз у реальному часі на відмінно від розробленого методу.

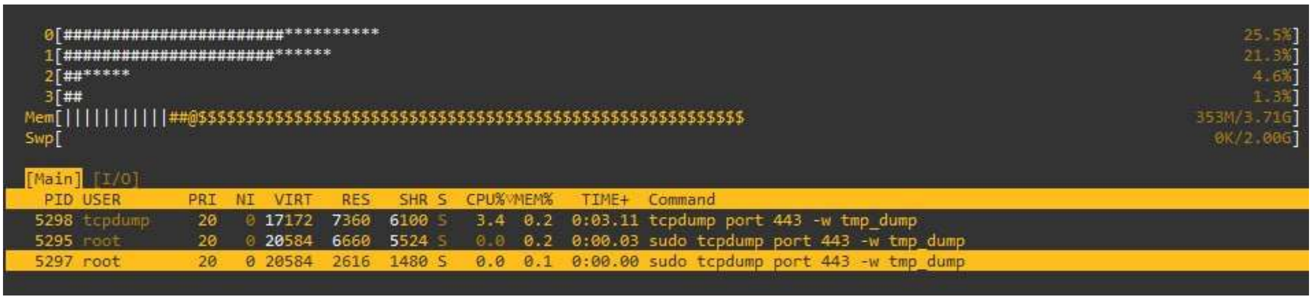


Рисунок 4.5 – Процес виміру використання ресурсів при зеркалювання трафіку за допомогою tcpdump

Провівши загальне порівняння запропонованого методу із реалізаціями існуючих методів, була побудована порівняльна діаграма, рис. 4.6.

Було проведене порівняння споживання ресурсів операційної системи одноплатного комп’ютера.

Для порівняння були обрані інструменти Wireshark/tcpdump, ntopng, NetFlow, які працюють на основі існуючих методів доступних на ринку рішень. Не всі аналізатори базуються на технологій eBPF та мають однаковий набір метрик які фіксуються, однак порівняння використання процесорного часу та кількість використаної оперативної пам’ять одразу продемонструвало суттєву різницю при виконанні на одноплатному комп’ютері.

Wireshark (tshark) продемонстурвав найвище споживання ресурсів: 85% ресурсів процесора та 520 МБ оперативної пам’яті. Це пояснюється тим, що libpcap копіював кожен пакет повністю із простору ядра у простір користувача, включаючи корисне навантаження, що створило значне навантаження на процесор та використало значний обсягу пам’яті для зберігання вмісту пакетів.

Ntopng продемонстрував використання 65% ресурсів процесора та 380 МБ оперативної пам’яті. Засіб також використовував libpcap для захоплення пакетів,

додатково виконував глибокий аналіз протоколів (DPI) через бібліотеку nDPI, що додало обчислювального навантаження. Значне споживання пам’яті обумовлене зберіганням таблиць потоків, результатів DPI та веб-інтерфейсу.

Запропонований метод використав 48% ресурсів процесора та 94 МБ оперативної пам’яті. Перевага в ресурсоефективності досяглась завдяки фільтрації та аналізу заголовків безпосередньо у просторі ядра без копіювання корисного навантаження пакетів. У простір користувача були передані лише структуровані події, розмір яких у десятки разів менший за повний пакет.

NetFlow агент продемонстрував найменше споживання: 10% ресурсів процесора та 45 МБ оперативної пам’яті. Однак цей засіб надав лише агреговану статистику потоків без детальних метрик якості з’єднання. За допомогою NetFlow було неможливо виміряти RTT, затримки, виявити повторні передачі або оцінити стан TCP-сесій, що обмежило його діагностичну цінність.

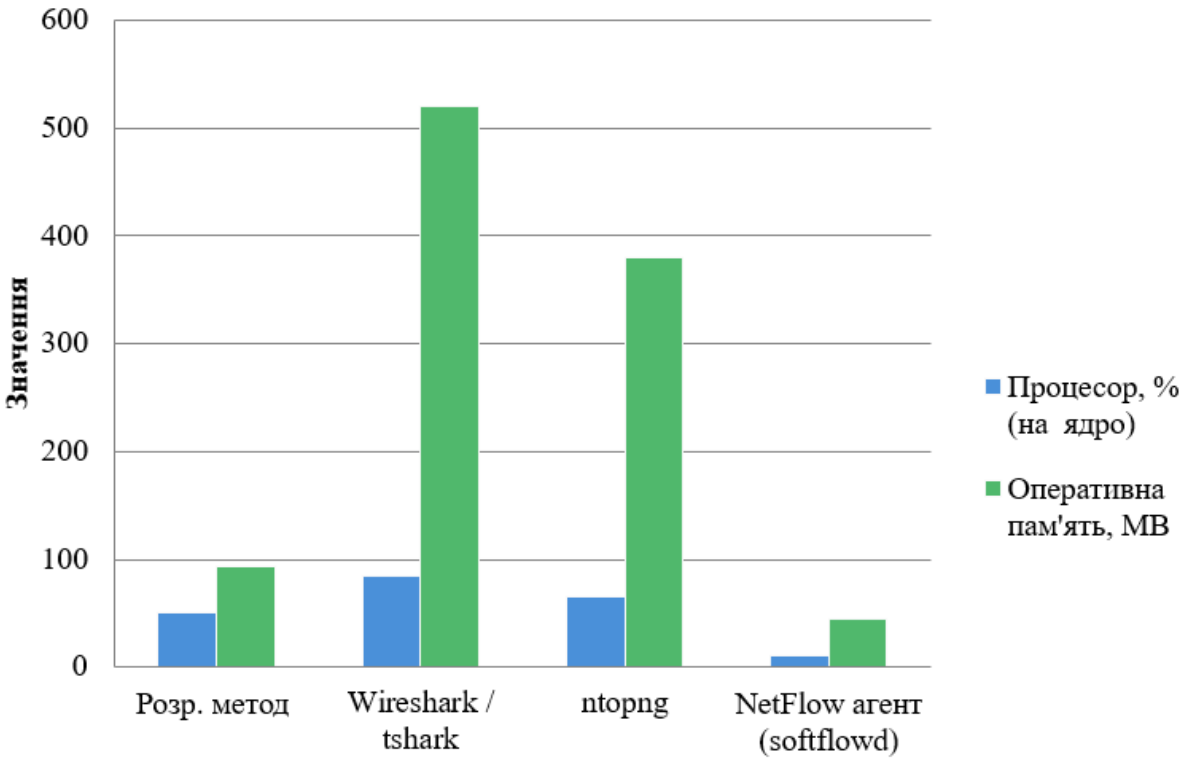


Рисунок 4.6 – Порівняльна діаграма запропонованого методу із існуючими реалізаціями методів

4.2 Тестування роботи методу і реалізації системи

На даному етапі було проведено тестування, в межах якого порівнювались фактичні виміри із вимогами які були визначені попередньо. Фазу тестування можна умовно розподілити на перевірку функційних вимог, тобто перевірити чи правильно працюють всі необхідні частини логіки, наступний крок – перевірка нефункційних вимог, тобто вимір і порівняння застосування системних ресурсів при використанні аналізатора та при звичайній роботі системи.

Для проведення вимірів застосовуються як системні утиліти операційної системи Linux, так і засоби маршрутизатора, щоб звірити отримані дані і порівняти їх із тими, які зібрав мережевий аналізатор.

Середовище проведення тестування є наступне: метод і реалізація виконуються на однопалатному комп'ютері Raspberry Pi 4B, порт в напрямку мережі включений у маршрутизатор Mikrotik hap ac3 та має швидкість передачі даних у розмірі 1000Мб/с, повний дуплексний режим, швидкість визначена механізмом автоузгодження, що продемонстровано на рис. 4.7.

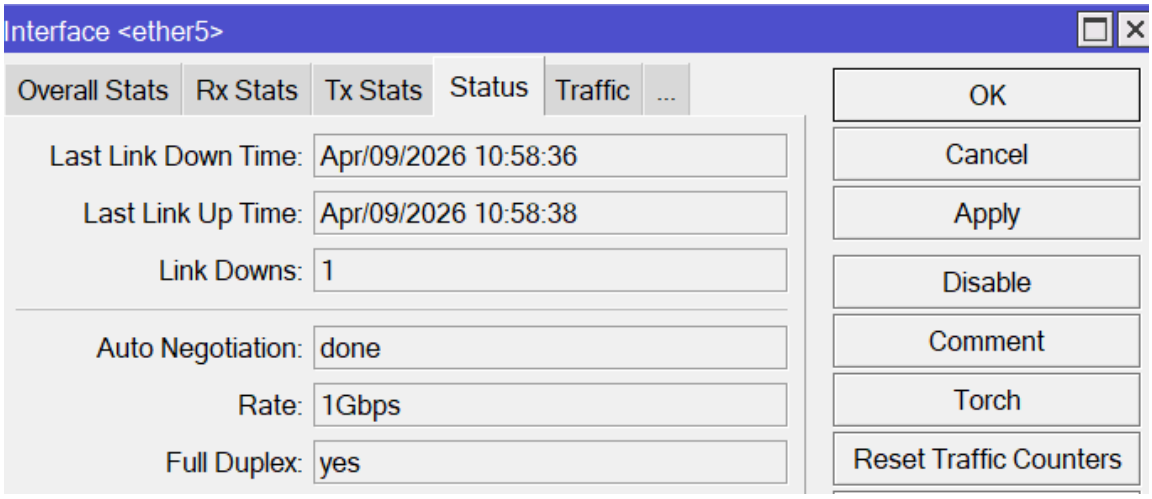


Рисунок 4.7 – Параметри фізичного порта через який відбувається тестування

Було проведено виміри затримок при передачі сегментів потоку, результати подані на рис. 4.8.

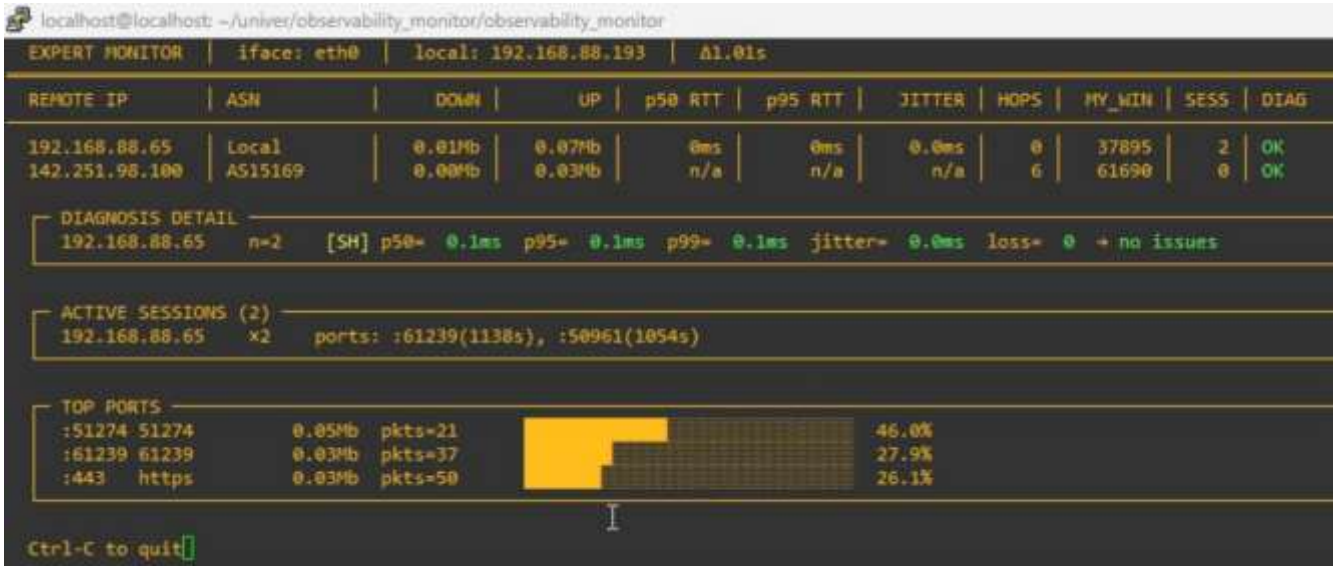


Рисунок 4.8 – Вимір затримок при передачі сегментів потоку

Запустивши утиліту mtr у одному вікні та аналізатор у іншому вікні, рис. 4.9, порівняємо значення.

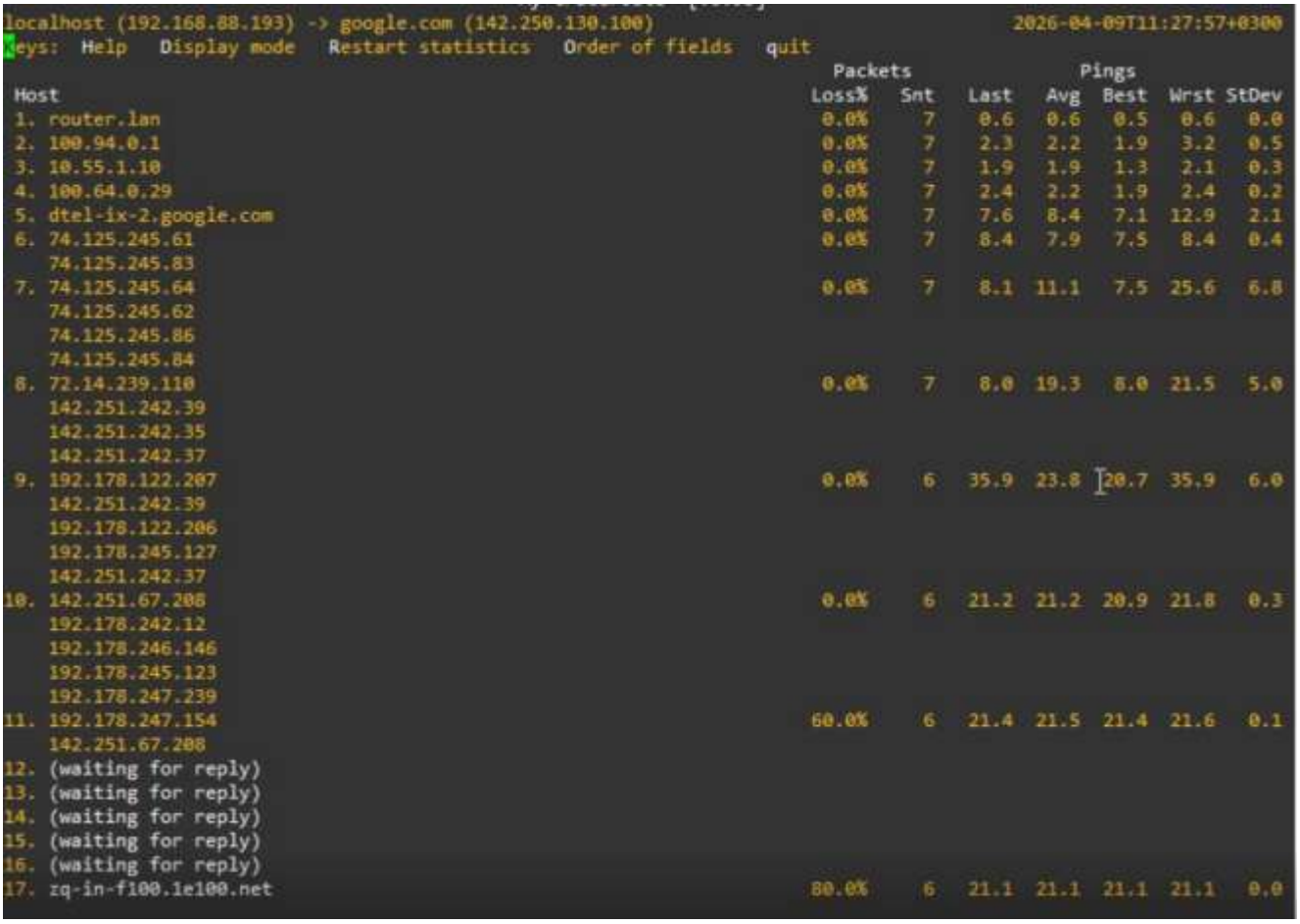


Рисунок 4.9 – Вимір затримок передачі утилітою MTR

Різниця між значеннями RTT менше 2%, значення затримок передачі сегментів потоку 0.2 мс. Мінімальна різниця в значеннях – це похибка, яка полягає у особливостях роботи методів, оскільки пасивний моніторинг не генерує трафіку, а лише аналізує існуючий, мережевий стек опрацьовує вхідні та вихідні пакети з різними пріоритетами відповідно.

Наступний крок – перевірити швидкість і якість реакції на мережеві втрати. Для цього запустимо аналізатор та штучно створимо втрати на пристрої за допомогою утиліти iptables, що є частиною мережевого екрану операційної системи Linux, із коефіцієнтом 0.3, тобто кожних 3 із 10 пакетів будуть відкидатись. Даний експеримент проведений, щоб перевірити на скільки швидко втрати відобразять наш метод та чи коректна діагностична інформація буде продемонстрована. Результат подано на рис. 4.10.

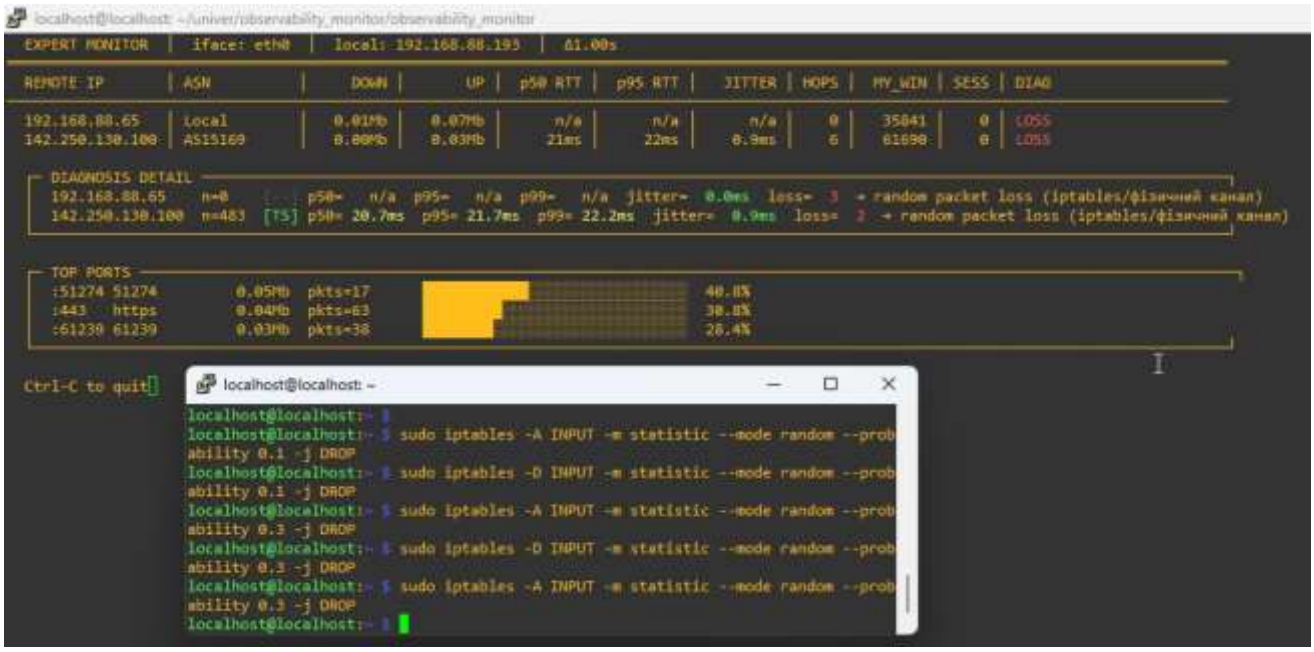


Рисунок 4.10 – Перевірка виявлення втрат сегментів потоку

Як можна побачити, після того як розпочались відомі і очікувані для нас втрати на шляху, засіб моніторингу просигналізував діагностичною інформацію у вигляді “DIAG: LOSS”, що розпочались втрати. Також було виведено інформацію адміністратору про можливу проблему із випадковими втратами на шляху, або не коректно налаштованим мережевим екраном. На основі результатів тест можна

вважати успішним. Після проведення експерименту правило мережевого екрану було знято.

Наступний етап – проведення експерименту для перевірки роботи методу у випадку, коли при завантаженні файлів великого об’єму кінцевий пристрій скидає розмір вікна передачі даних. При розвантаженні мережевого буферу вікно знову збільшується до точки, коли буфер не стане переповнений. Така поведінка пристрою зазвичай відбувається через неможливість обробляти той об’єм даних, який передається через мережеві інтерфейси. Проведемо дослід, результати якого зображенні на рис. 4.11.

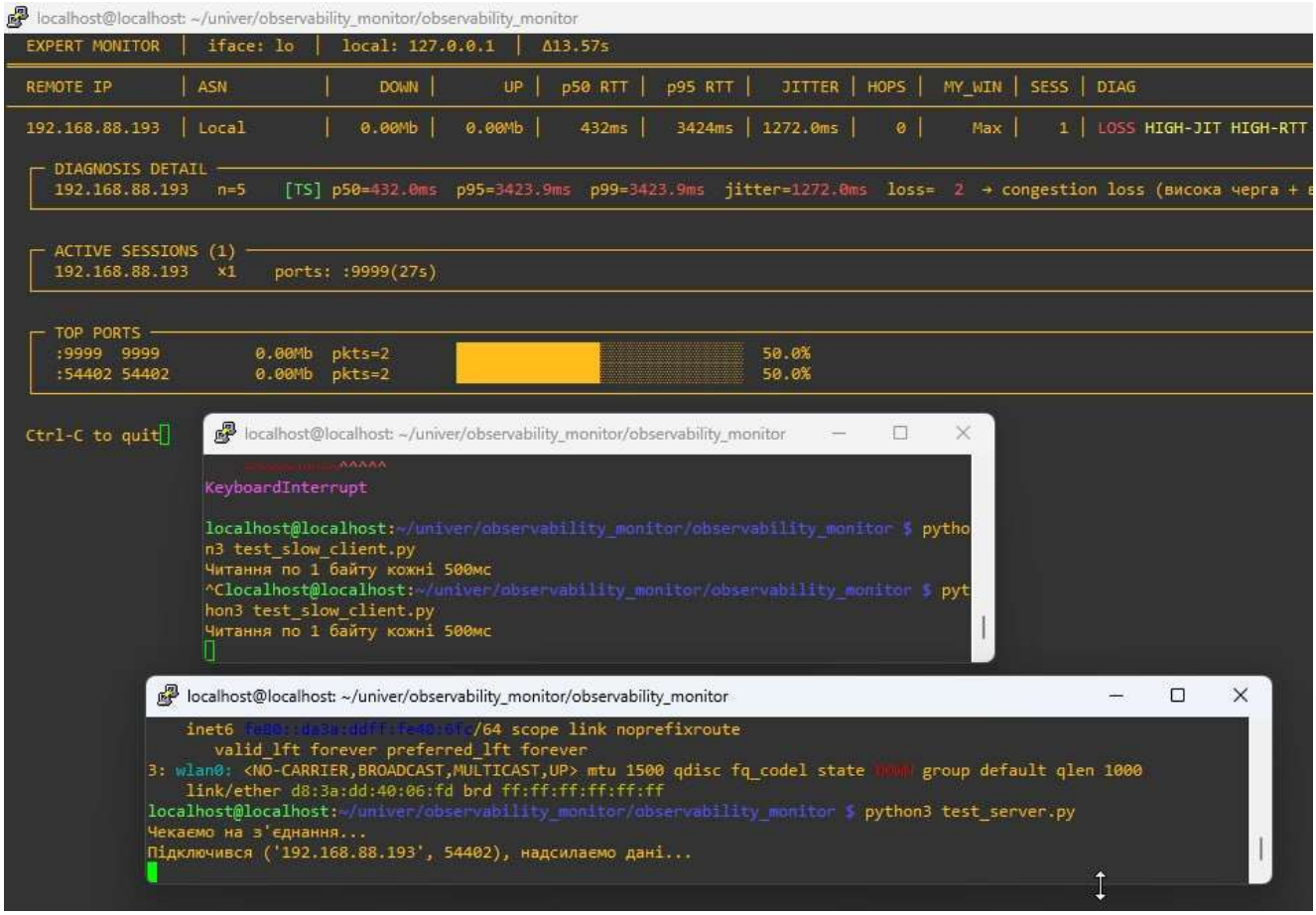


Рисунок 4.11 – Перевірка виявлення втрат сегментів потоку

Для експерименту було створено спеціальну симуляцію сервера та повільного клієнта за допомогою Python коду. Сервер працює без обмежень, а скрипт клієнта симулює штучне сповільнення передачі даних на TCP порту 54402, читаючи по байту кожні 500мс, знижуючи при цьому швидкість передачі.

Як можна побачити на попередньому зображенні, метод спрацював коректно і відреагував на затримки, вказуючи на можливу проблему на мережевому рівні пристрою, який здійснює комунікацію. Повідомлення інформують про втрати, аномалію із затримками, надто велику різницю між двома ітераціями комунікації, цього може бути достатньо, щоб розпочати більш глибоке відстеження проблеми.

Досить не стандартним способом була виявлена проблема із пристроєм на якому проводились тести. При завантаженні файлу значного об’єму, близько 1ГБ із віддаленого ресурсу, періодично спостерігалось різке зниження швидкості.

Початково система моніторингу не виявляла жодних втрат сегментів потоку, однак при рості швидкості завантаження файлу було виявлено діагностичне повідомлення типу “PC-OVERLD”, яке сигналізувало про те що об’єм мережевого буферу вичерпаний, що подано на рис. 4.12.

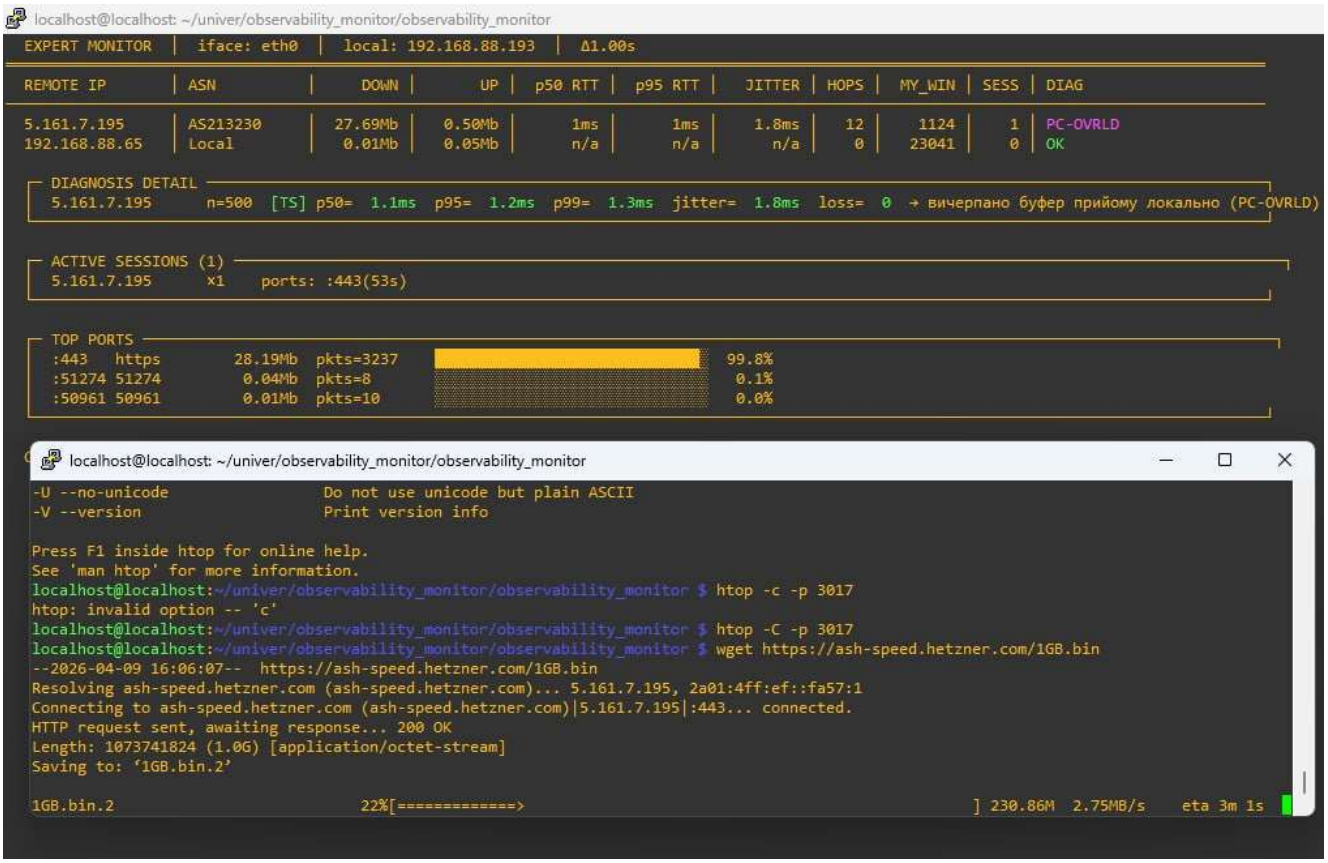


Рисунок 4.12 – Виявлена аномалія

Ідентифікована аномалія стала причиною для більш детального пошуку першопричини проблеми. Тому переглянувши швидкість запису на накопичувач ідентифіковано проблему, яка і впливала на швидкість завантаження, а саме накопичувач, що зображено на рис. 4.13.

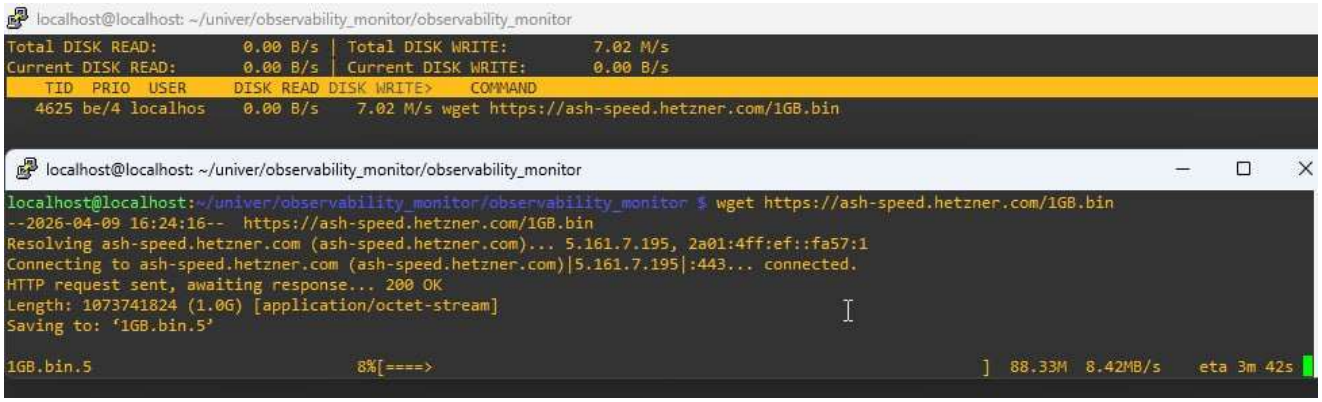


Рисунок 4.13 – Причина різкого зниженням швидкості завантаження

4.3 Інтеграція із Prometheus, нормалізація і публікація метрик

Однією із не функційних вимог до реалізації і впровадження методу є швидке впровадження рішення і уніфікація із існуючими системами клієнта. Для того, щоб швидко і якісно передавати метрики і інтегрувати їх у існуючи системи моніторингу, метрики мають бути зведені до стандартизованого формату.

Prometheus як сервіс є фактично стандартним методом збору і зберігання метрик, в продуктових системах клієнта він зустрічається надзвичайно часто через свою практичність, невибагливість середовища, швидкість роботи.

Модель даних, які може надавати Prometheus базується на часових рядах, а дані подані у форматі ключ – значення. Якщо застосувати Prometheus як вбудований модуль для реалізації нашого методу, то можливо вирішити декілька технічних проблем одночасно, а саме: постійне зберігання історичних метрик, можливість кращої візуалізації через побудову графіків у системах візуалізації типу Grafana або інших, що підтримують формат Prometheus. Зберігання у агрегованому форматі частково вирішує проблему відмовостійкості, оскільки навіть при збої

пристрою, на якому виконується сервіс, метрики будуть збереженні у постійній пам’яті. Для зручності можна також налаштувати автоматичне резервне копіювання.

Метод надає можливість вибору кінцевому користувачу, можливість налаштувати засіб максимально відносно своїх вподобань відповідно у сумісності до власної інфраструктури. Доступні корпоративні рішення на момент дослідження зазвичай є більш закритими до модифікацій основних функцій, досить складно знайти відкриті рішення, що надають свободу вибору для приватних клієнтів чи бізнесу.

Якщо розглянути представлення мережевих метрик у Prometheus, то всі зібрані метрики є структурованого формату ключ - значення, для прикладу “remote_ip=8.8.8.8, asn=15169, 12,4”, як приклад показаний набір, що стосується потоку до IP адреси 8.8.8.8, що знаходиться у автономній системі 15169, затримка RTT до якої є 12.4 мс. Технічно можливо впроваджувати нові додаткові поля, при оновленні, не порушуючи початкової структури і не створюючи проблем для користувачів.

Одним із етапів перед публікацією метрик є їх нормалізація, коли початково зібрані дані доопрацьовуються перед публікацією. Якщо уникнути процесу нормалізації, то може виникнути ситуація, коли значення технічно коректні, але статистично не є достовірними і створюють аномалії, коли проводиться х порівняння.

У просторі ядра вимірювання затримок при передачі фрагментів відбувається у наносекундах при виконанні функції bpf_ktime_get_ns(), більш доцільним є публікація у мілісекундах. Значення пропускної здатності початково вимірюється у байтах за одиницю часу, але для більшої інтуїтивності відбувається перехід до більш звичних одиниць виміру Мбіт/с. Переходи до іншої звичної більш стандартної одиниці виміру напряму впливає на спрощення інтерпретації вимірів на графіках.

Окремо було звернуто увагу на нормалізацію часу інтервалу. При обробці метрик, перенесені даних між просторами виникає похибка при вимірі тривалості

інтервалу. За стандартним підходом очікувався поділ на інтервали у вигляді однієї секунди, проте у пікові періоди, коли кількість трафіку є великою, то тривалість обробки є більшою і інтервал не є фіксованого розміру однієї секунди. Фактичний інтервал коливався від 1.00 до 1.05 секунд. Без коригувань і динамічного розрахунку значень інтервалу відбувалось створення аномалії вимірювання, затримки фрагментів потоків зростали на 5% від стандартних значень. Така поведінка системи є не допустимою і впливає на результат, створюючи видимість аномалії трафіку, якої фактично не існує. Тому параметр дельта часу було враховано при фіксувані метрик і також враховано при обчисленні швидкості передачі даних.

Важливим аспектом є перевірка значення затримок RTT на достовірність, порівнюючи їх із значеннями визначеними в умові. Нижня і верхня межі були визначенні у розділі формалізованого представлення системи у вигляді замкненої системи рівнянь.

Коли система намагається визначити і фіксувати значення автономної системи на потоках трафіку, це очевидно вимагає додаткового часу на виконання DNS запитів. Значення автономної системи не є особливо критичним, щоб не сповільнювати обробку наступних фрагментів потоків, то значення ASN тимчасово може записуватись як “asn=…”, після чого доповнюється справжнім значенням типу “asn=15169”, коли DNS запит виконався і отримано результат. Існують виключення, які необхідно врахувати початково - DNS запит не повернув результат, через певну кількість часу. Відповідно ситуація, коли у Prometheus буде опубліковане значення “asn=…” не є прийнятним, тому воно замінюється на значення типу “asn=unknown”.

Щодо формату публікації метрик, то вибраний формат підтримується версіями Prometheus новішими за версію 2.0. Для зручності читання додано рядки коментарів, що допомагають зрозуміти опис значення. Метрики публікуються за http/https шляхом /metrics оновлюються автоматично, функція кешування вимкнена. За замовчуванням Prometheus збирає і публікує метрики один раз у 15

секунд, для мережевого моніторингу цього не достатньо, тому конфігурація змінена і інтервал скорочений до 5 секунд.

Вигляд опублікованих метрик зображено на рис. 4.14.

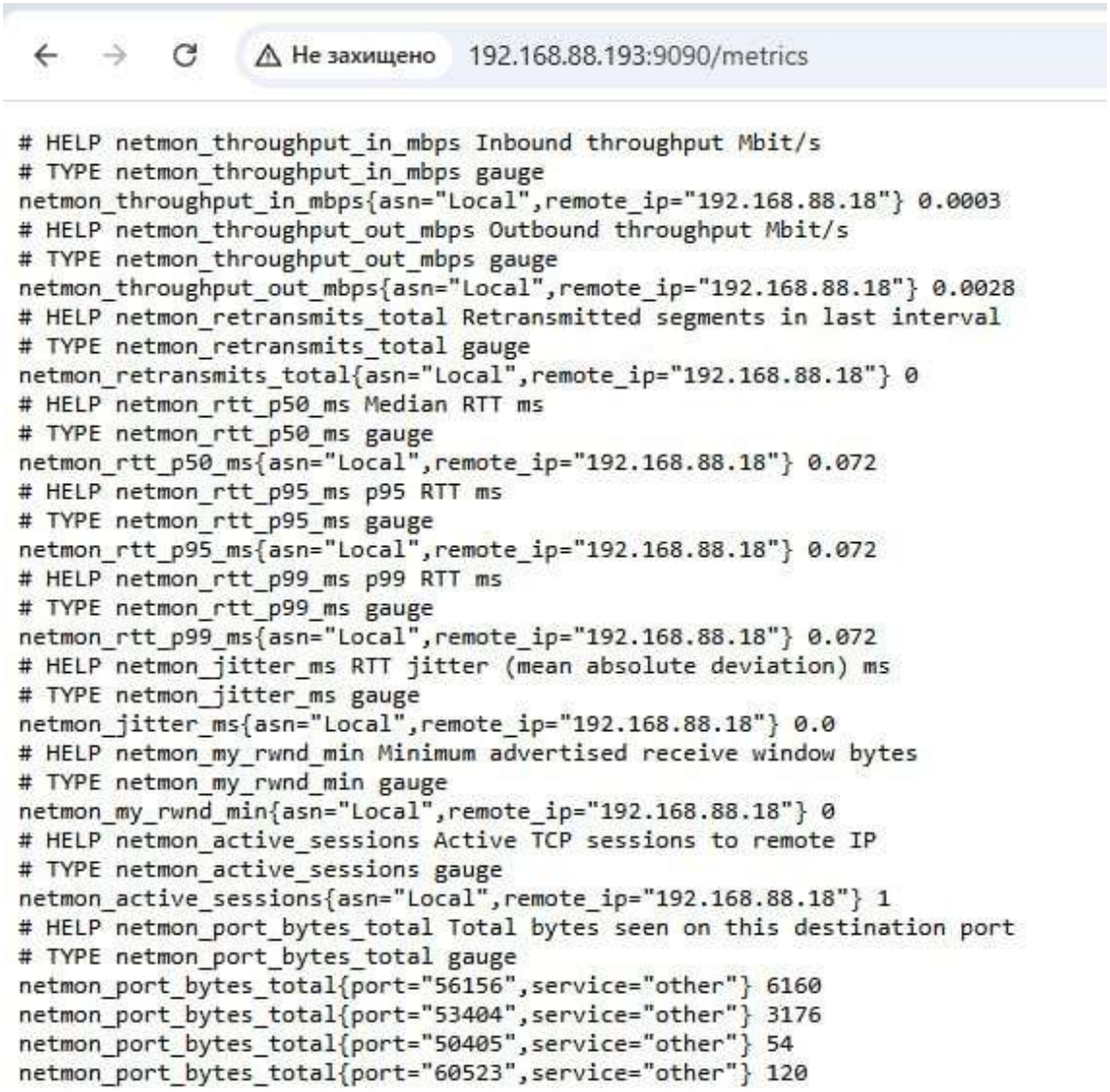


Рисунок 4.14 – Вигляд опублікованих метрик Prometheus, /metrics

4.4 Програмна реалізація

Програмна реалізація методу пасивного моніторингу складається з трьох основних компонентів. Файл bpf/program.c - це частина реалізації, яка містить програму мовою програмування C, що компілюється і виконується саме у просторі ядра Linux, механізмом eBPF. Файл monitor.py - це головний процес в просторі

користувача, написаний на мові програмування Python. Файл містить реалізацію компонентів обробки подій, агрегацію метрик, діагностику, логування та демонстрацію стандартними засобами вводу-виводу. Файл prometheus.py - це фактично модуль HTTP сервера для публікації метрик. Реалізована структура програмних компонентів при реалізації спрямована на їх розмежування і підвищення відмовостійкості.

Файл program.c завантажується після запуску основної частини логіки. Щоб виконати цю частину коду, необхідно. щоб відбулась компіляція і верифікація описаною eBPF логіки. Компіляція коду на даному етапі відбувається за допомогою бібліотеки BCC з використанням LLVM/Clang.

Таким чином файли, що містять логіку реалізації мають наступне призначення:

Таблиця 4.1 – Основні файл для реалізації та їх призначення

Середовище	Файл	Частина логіки
Ядро Linux	bpf/program.c	Збір потоків, виявлення втрат, перевірка стану з'єднань, основна логіка методу
Користувацький простір, Python	monitor.py	Агрегація метрик, діагностика, вивід результатів
Користувацький простір, Python	prometheus.py	HTTP сервер, реєстр метрик та місце де вони публікуються

Якщо розглянути більш детально програму у просторі ядра, то точкою входу є функція analyze_tcp, що прикріплюється до сокету мережевого інтерфейсу через виклик функції b.attach_raw_socket. Вона викликається ядром для кожного TCP-пакету, що проходить через інтерфейс. Читання полів заголовків IP і TCP виконується через функції load_byte, load_half, load_word за абсолютними зміщеннями від початку пакету.

Для зберігання стану між викликами використовуються три BPF мапи типу BPF_HASH. Карта entry_timestamp зберігає часову мітку ядра при отриманні

фрагменту із значенням SYN для подальшого обчислення затримки нашим методом, при надходженні фрагменту із значенням SYN-ACK.

Аналіз значення, що відповідає за часову мітку в фрагменті TCP реалізований через макрос _TSOPT_STEP, що розгортається у 20 послідовних незалежних блоків умов. Це рішення не є ідеальним із точки зору класичного програмування, проте є обов’язковою вимогою верифікатора eBPF при перевірці синтаксису, цикл з конструкцією break відхиляється верифікатором, оскільки умова виходу залежить від вмісту пакету, що не є допустимою конструкцією.

Результати обробки передаються у простір користувача через два окремих буфери типу BPF_PERF_OUTPUT: events для подій, що стосуються фрагментів (~60 байт на подію) і conn_events для подій зміни стану TCP з'єднань (SYN, SYN-ACK, FIN, RST). Розділення на два буфери необхідне через різну частоту подій: події відносно сегментів генеруються для кожного пакету, а події стану лише при встановленні і завершенні з'єднань.

Якщо більш детально розглянути частину в користувацькому просторі, то передача даних від ядра в користувацький простір відбувається у циклі кожні 100 наносекунд часу. Всі виклики відбуваються саме в цьому потоці, завдяки такому рішенню основні структури даних, що передаються, не потребують окремої синхронізації.

Визначення локальних IP-адрес виконується через системний виклик SIOCGIFCONF з функцією fcntl.ioctl, що повертає таблицю адрес ядра для вказаного інтерфейсу. Результат зберігається у множині LOCAL_IPS і приналежність IP адреси до неї перевіряється операцією складності O(1) для кожного пакету.

Таймер інтервалу реалізований без окремого потоку, а через метод check_timer викликається на кожній пакетній події і перевіряє час - last_print ≥ 1.0. При спрацюванні виконується tick(delta) де delta є реальним часом інтервалу, тобто значенням, що не є фіксованою 1 секундою. Як було зазначено попередньо при створенні методу, адаптивний часовий інтервал потрібен для коректної нормалізації пропускної здатності і щоб уникнути штучних аномалій при вимірах.

4.5 Висновки

В розділі 4 було проведено ряд вимірів для з’ясування точності роботи системи, також були проведені виміри, спрямовані на фіксацію кількості системних ресурсів, які необхідні для того, щоб метод та його реалізація функціонували коректно.

Як частину архітектурного рішення було продемонстровано можливість інтеграції із Prometheus та процес публікації метрик.

Експерементальне дослідження було проведено в умовах мережевого навантаження при завантаженні файлу, це було спрямовано на перевірку роботи методу при інтенсивному трафіку, проходженні TCP-трафіку через мережевий інтерфейс.

За результатами вимірювання було встановлено відповідність метрик у порівнянні із тими, що були запропоновані як оптимальні. Порівняння із існуючими методами продемонструвало споживання на 26% менше процесорного часу та на 45% менше використання оперативної пам’яті.

Отримані результати демонструють придатність розробленого методу до використання на пристроях із обмеженою кількістю ресурсів, зокрема на однопалатних комп’ютерах при цьому не впливаючи негативно на транзитний мережевий трафік.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто проблему мережевого моніторингу та виявлення мережевих аномалій, за результатами виконаних теоретичних та практичних досліджень розроблено метод моніторингу параметрів комп’ютерних мереж на основі аналізу мережевих з’єднань, що забезпечує збирання, обробку та візуалізацію метрик якості з’єднання в реальному часі при мінімальному споживанні системних ресурсів в реальному часі.

Проведений порівняльний аналіз активних та пасивних засобів моніторингу виявив, що пасивний аналіз реального трафіку є більш точним, оскільки не створює додаткового навантаження на канали зв'язку та не спотворює вибірку синтетичними даними. Водночас визначено ключові виклики такого підходу, зокрема необхідність обробки великих обсягів інформації безпосередньо «на льоту» та складність відновлення стану TCP-сесій.

Запропонований метод моніторингу базується на використанні технології eBPF, що дозволяє виконувати фільтрацію та агрегацію метрик безпосередньо у просторі ядра операційної системи. Це вирішує проблему надмірної надлишковості та виснаження ресурсів процесора, притаманну класичним інструментам, які копіюють повні пакети у користувацький простір.

Математична формалізація методу охоплює ідентифікацію потоків за набором ключових параметрів та вимірювання RTT двома способами: через аналіз станів SYN/SYN-ACK та за допомогою опції TCP Timestamp. Такий комбінований підхід забезпечує високу точність вимірювань навіть у складних умовах перевпорядкування пакетів або відсутності підтримки певних опцій мережевим обладнанням.

Програмна реалізація методу, що поєднує мови C та Python, демонструє ефективний розподіл задач: низькорівнева обробка трафіку відбувається у просторі ядра, а складна агрегація та взаємодія з сервером метрик – у просторі користувача. Це дозволяє мінімізувати копіювання даних між просторами, передаючи лише

значущі поля пакетів, що критично важливо для пристроїв з обмеженими ресурсами.

Набула подальшого розвитку інформаційна технологія пасивного моніторингу мережевих потоків на основі запропонованого методу, практична реалізація якого інтегрується з існуючими системами збору метрик Prometheus, Grafana та може бути розгорнута на пристроях з обмеженими ресурсами, зокрема одноплатних комп'ютерах.

Впровадження результатів роботи підтверджують, що впровадження інтелектуальної діагностики аномалій на основі зібраних метрик дозволяє не лише констатувати стан мережі, а й оперативно виявляти причини збоїв, такі як перевантаження буферів чи нестабільність маршрутів. Запропонований підхід забезпечує приріст продуктивності та надійність керування сучасною мережевою інфраструктурою за рахунок пасивного моніторингу параметрів комп'ютерних мереж із використанням технології eBPF, що забезпечує вимірювання затримок (RTT), виявлення повторних передач, оцінку кількості проміжних вузлів та аналіз стану TCP-сесій в реальному часі.

Поставлену мету було досягнуто шляхом розв'язання таких основних завдань:

- проаналізовано існуючі методи та засоби моніторингу параметрів комп'ютерних мереж, визначено їх переваги та недоліки;

- розроблено математичну модель ідентифікації мережевих потоків та формалізовано метрики якості з'єднання у вигляді замкненої системи рівнянь;

- розроблено метод пасивного моніторингу на основі eBPF-фільтрів з декомпозицією на підсистеми захоплення, обробки та публікації метрик;

- здійснено програмну реалізацію розробленого методу та проведено експериментальне дослідження споживання ресурсів і точності вимірювань в умовах реального мережевого навантаження.

За темою кваліфікаційної роботи опубліковано одну публікацію [75] у науковому журналі «Вимірювальна та обчислювальна техніка в технологічних процесах» (Хмельницький – 2026). С. 49-58.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Зубенко І., Варпіоте І., Петричук Д. Перенесення офлайн-бізнесу в онлайн. *Наукові записки Національного університету «Острозька академія». Серія: Економіка*. 2025. Вип. 36 (64). С. 31–35. URL: https://doi.org/10.25264/2311-5149-2025-36(64)-31-35.

2. Akinade A. O., Adepoju P. A., Ige A. B., Afolabi A. I. Cloud security challenges and solutions: a review of current best practices. *International Journal of Multidisciplinary Research and Growth Evaluation*. 2025. Vol. 6, no. 1. P. 26–35. URL: https://doi.org/10.54660/.IJMRGE.2025.6.1.26-35.

3. Kirti M. et al. Fault-tolerance approaches for distributed and cloud computing environments: a systematic review, taxonomy and future directions. *Concurrency and Computation: Practice and Experience*. 2024. URL: https://doi.org/10.1002/cpe.8081.

4. Liu Y. et al. Enhancing critical network infrastructure resilience through optimal post-disruption maintenance and routing decisions. *Reliability Engineering & System Safety*. 2024. URL: https://doi.org/10.1016/j.ress.2024.110717.

5. Sharma R. et al. AutoML-enabled infrastructure for adaptive personalization in high-performance e-commerce systems. *International Journal of Artificial Intelligence, Big Data, Computing and Data Management Systems*. 2026. URL: https://doi.org/10.XXXX/ijaibdcms.2026.372.

6. Min S., Kim B. AI technology adoption in corporate IT network operations based on the TOE model. *Digital*. 2024. Vol. 4, no. 4. P. 947–970. URL: https://doi.org/10.3390/digital4040047.

7. Buhari A., Athithan A. Influence of Internet and its connectivity in workplace - a comprehensive analysis. *Recent Research Reviews Journal*. 2024. Vol. 3, no. 1. P. 244–257. URL: https://doi.org/10.36548/rrrj.2024.1.016.

8. Remote work statistics and trends for 2025. *Robert Half*. 2025. URL: https://www.roberthalf.com/us/en/insights/research/remote-work-statistics-and-trends (дата звернення: 21.04.2026).

9. A new static cost-effective parameter for interconnection networks of massively parallel computer systems / M. M. Hafizur Rahman et al. *Soft Computing in Data Analytics*. Singapore, 2018. P. 147–155. URL: https://doi.org/10.1007/978-981-13-0514-6_15.

10. Tahlyan D., Mahmassani H., Stathopoulos A. et al. In-person, hybrid or remote? Employers' perspectives on the future of work post-pandemic. *arXiv preprint*. 2024. URL: https://doi.org/10.48550/arXiv.2402.18459.

11. Method for estimating the convergence parameters of dynamic routing protocols in computer networks / H. Osukhivska et al. *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 22–25 September 2021. 2021. URL: https://doi.org/10.1109/csit52700.2021.9648792.

12. Huang P., Guo C., Zhou L., Lorch J. R., Dang Y., Chintalapati M., Yao R. Gray failure: the Achilles' heel of cloud-scale systems. *Hot Topics in Operating Systems (HotOS XVII)*. 2017. URL: https://doi.org/10.1145/3102980.3103005 (дата звернення: 21.04.2026).

13. Shete P. J., Awale R. N., Ket S. Y. Channel quality aware cross-layer design based rate adaptive MAC for improving the throughput capacity of multi-hop ad hoc networks. *Ad Hoc Networks*. 2017. Vol. 63. P. 45–61. URL: https://doi.org/10.1016/j.adhoc.2017.05.009.

14. Fabris M., Ceccato R., Zanella A. Efficient sensors selection for traffic flow monitoring: an overview of model-based techniques leveraging network observability. *Sensors*. 2025. Vol. 25, no. 5. Article 1416.

15. Kaur P., Gill N. Performance comparison of UDP and UDP-Lite for different video codecs. *International Journal of Computer Applications*. 2012. Vol. 54, no. 12. P. 15–22. URL: https://doi.org/10.5120/8617-2479.

16. Avasthi S., Vishwakarma S. K. Network issues and high-level communication tools in distributed computing. *Distributed and Parallel Computing*. 2025. P. 115–134.

17. Mansurov T., Zhdanovich S., Mansurov E. A distributed telecommunication network monitoring system model. *Agricultural Sciences*. P. 87.

18. Avasthi S., Vishwakarma S. K. Network issues and high-level communication tools in distributed computing. *Distributed and Parallel Computing*. 2025. P. 115–134.

19. Kamtam A., Kamar A., Patkar U. C. Artificial intelligence approaches in cyber security. *International Journal on Recent and Innovation Trends in Computing and Communication*. 2016. Vol. 4, iss. 4. P. 5–9.

20. Kovalenko O. Y., Kuzniuk K. V. Computer network monitoring systems. *Mathematical Machines and Systems*. 2023. Vol. 1. P. 50–59. URL: https://doi.org/10.34121/1028-9763-2023-1-50-59.

21. Wang C., Zhang X., Lu R., Lin X., Zeng X., Zhang X. et al. Towards LLM-based failure localization in production-scale networks. *Proceedings of the ACM SIGCOMM 2025 Conference*. 2025. September. P. 496–511.

22. Pimenta Junior A. P., Abe J. M., Silva G. C. Determination of operating parameters and performance analysis of computer networks with paraconsistent annotated evidential logic Eτ. *IFIP Advances in Information and Communication Technology*. Cham, 2016. P. 3–11. URL: https://doi.org/10.1007/978-3-319-51133-7_1.

23. Cloudflare outage on November 18, 2025. *Cloudflare Blog*. 2025. URL: https://blog.cloudflare.com/18-november-2025-outage/ (дата звернення: 21.04.2026).

24. Raj P. Understanding the October 2025 AWS US-East-1 outage: a deep developer-level postmortem. *Medium*. 2025. URL: https://medium.com/@therajpiyush/understanding-the-october-2025-aws-us-east-1-outage-a-deep-developer-level-postmortem-d13cbdac6cc8 (дата звернення: 21.04.2026).

25. Kovvuru I. Microsoft Azure outage Oct 29, 2025: root cause, impact and technical analysis. *Medium*. 2025. URL: https://medium.com/@ismailkovvuru/microsoft-azure-outage-oct-29-2025-root-cause-impact-and-technical-analysis-3c7646d31703 (дата звернення: 21.04.2026).

26. Martinho C., Strickx T. Understanding how Facebook disappeared from the Internet. *Cloudflare Blog*. 2021. URL: https://blog.cloudflare.com/october-2021-facebook-outage/ (дата звернення: 21.04.2026).

27. Strickx T., Levy M. How Verizon and a BGP optimizer knocked large parts of the Internet offline today. *Cloudflare Blog*. 2019. URL: https://blog.cloudflare.com/how-verizon-and-a-bgp-optimizer-knocked-large-parts-of-the-internet-offline-today/ (дата звернення: 21.04.2026).

28. Ozdem M. A novel approach for real-time anomaly detection in dynamic computer networks using temporal graph networks and explainable artificial intelligence. *Alexandria Engineering Journal*. 2025. Vol. 132. P. 369–382.

29. Рибачок Я. Аналіз методів для виявлення мережевих аномалій. *ББК 32.97 Е50*. 2025. С. 279. URL: http://repositsc.nuczu.edu.ua/bitstream/123456789/27149/1/%D0%9C%D0%B0%D0%BA%D0%B5%D1%82%202025.pdf#page=280.

30. Середа О. В. Програмне забезпечення системи спеціалізованих внутрішніх міжсегментних мережних екранів ISFW. 2025. URL: https://dspace.kntu.kr.ua/items/fa37bd7d-298e-4285-b406-0d2c24018957 (дата звернення: 21.04.2026).

31. Куренков Б. М. Мікросервісна архітектура IT-сервісу обслуговування замовлень. 2025. URL: https://openarchive.nure.ua/entities/publication/bb645a32-6897-474d-a7c0-7e8b6170584b (дата звернення: 21.04.2026).

32. Бабич С. М., Ляшук Т. Г. Еволюція архітектур програмного забезпечення обчислювальних систем: від монолітних до мікросервісних. *Інфокомунікаційні та комп'ютерні технології*. 2025. № 2 (10). С. 30–36. URL: https://visn-icct.uu.edu.ua/index.php/icct/article/view/195.

33. Сікора Р. В. Domain Driven Design як основа для моделювання мікросервісів в електронній комерції. *Таврійський науковий вісник. Серія: Технічні науки*. 2025. № 1. С. 203–209. URL: http://www.journals.ksauniv.ks.ua/index.php/tech/article/download/793/737.

34. Bonagiri V., Bhatnagar A., Nethi R. Event-driven API-based architecture supporting real-time distributed databases. *International Journal of Data Science and IoT Management System*. 2025. Vol. 4. P. 544–555. URL: https://doi.org/10.64751/ijdim.2025.v4.n4.pp544-555.

35. Choudhary S. K. Implementing event-driven architecture for real-time data integration in cloud environments. *International Journal of Computer Engineering & Technology*. 2025. Vol. 16, no. 1. P. 1535–1552. URL: https://www.researchgate.net/profile/Siddharth-Choudhary-5/publication/388534188_Implementing_Event-Driven_Architecture_for_Real-Time_Data_Integration_in_Cloud_Environments.

36. Bruccoleri R., Russo M., Smith A., Chasalow S. Docker Image Builder: a robust method for constructing Docker images for reproducible research. *OSF*. 2025. URL: https://osf.io/download/8pgd7.

37. Koneru N. Containerization best practices - using Docker and Kubernetes for enterprise applications. *Journal of Information Systems Engineering and Management*. 2025. Vol. 10. P. 724–746. URL: https://doi.org/10.52783/jisem.v10i45s.8905.

38. Dugbartey A. N., Kehinde O. Optimizing project delivery through agile methodologies: balancing speed, collaboration and stakeholder engagement. *World Journal of Advanced Research and Reviews*. 2025. Vol. 25, no. 1. P. 1237–1257.

39. Weijers-Coghlan J., Wallis L., Burnham R., Kotlarewski N. Attempting to define design-build education in architecture. *Archnet-IJAR: International Journal of Architectural Research*. 2026. P. 1–20. URL: https://doi.org/10.1108/ARCH-10-2025-0476.

40. Augustyniak P., Nierbiński M., Zwierzykowski P. Identification of characteristics of Rouge Access Point based on SNMP and SYSLOG protocols. *Journal of Eastern Europe Research in Business and Economics*. 2026. URL: https://doi.org/10.5171/2025.4531525.

41. Elkin R., Oh J. H., Simhal A., Deasy J. Abstract 5520: multi-modal integration in the NetFlow framework for comprehensive exploratory data analysis. *Cancer Research*. 2026. Vol. 86. P. 5520. URL: https://doi.org/10.1158/1538-7445.AM2026-5520.

42. Gbigbidje F., Oghuvwu B., Victor O., Ewere O., Alele J., Obayehagweme E. Evaluating latency and packet loss of direct and hotspot Internet connections for quality

of service and network selection. *International Journal of Recent Engineering Science*. 2025. Vol. 12. P. 34–49. URL: https://doi.org/10.14445/23497157/IJRES-V12I4P104.

43. Northrop J. Monitoring and observability in cloud systems. 2026. URL: https://www.researchgate.net/publication/401344570_Monitoring_and_Observability_in_Cloud_Systems (дата звернення: 21.04.2026).

44. Wang S., Jin T., Chen C., Ma Y., Xiaojing W., Guan X. Observability guarantee in distributed edge sensing for industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics*. 2026. P. 1–12. URL: https://doi.org/10.1109/TII.2026.3665608.

45. Christopher A., Cox G., Pum M. Monitoring serverless applications in AWS and Azure. 2024. URL: https://www.researchgate.net/publication/391018431_Monitoring_Serverless_Applications_in_AWS_and_Azure (дата звернення: 21.04.2026).

46. Sanepalli U. R. Architecting multi-region observability in AWS: a hybrid framework using CloudWatch, Prometheus, and Grafana. *International Journal for Multidisciplinary Research*. 2025. Vol. 7. URL: https://doi.org/10.36948/ijfmr.2025.v07i05.69008.

47. Montanari A., Duan C., Motter A. Target controllability and target observability of structured network systems. *IEEE Control Systems Letters*. 2023. Vol. 7. P. 3060–3065. URL: https://doi.org/10.1109/LCSYS.2023.3289827.

48. Analysis of network parameters influencing performance of hybrid multimedia networks / D. Kovac et al. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*. 2013. Vol. 2, no. 3. URL: https://doi.org/10.11601/ijates.v2i3.69.

49. Kaprakattu A. R. Intelligent health monitoring and adaptive restart mechanism for containerized network functions. *European Journal of Computer Science and Information Technology*. 2025. Vol. 13. P. 91–108. URL: https://doi.org/10.37745/ejcsit.2013/vol13n2891108.

50. Журило О., Ляшенко О., Аветісова К. Огляд рішень з апаратної безпеки кінцевих пристроїв туманних обчислень у інтернеті речей. *Сучасний стан наукових досліджень та технологій в промисловості*. 2023. № 1 (23). С. 57–81.

51. Pattan K. K. Dynamic memory optimization in containerized environments through machine. 2025.

52. Hassan M. Software architecture between monolithic and microservices approach. *SSRN Electronic Journal*. 2024. URL: https://doi.org/10.2139/ssrn.4753649.

53. Slivka S. Microservices architecture for ERP systems. *Bulletin of Cherkasy State Technological University*. 2024. URL: https://doi.org/10.62660/bcstu/4.2024.32.

54. Faseeha U., Syed H. J., Samad F., Zehra S., Ahmed H. Observability in microservices: an in-depth exploration of frameworks, challenges, and deployment paradigms. *IEEE Access*. 2025. URL: https://doi.org/10.1109/ACCESS.2025.3562125.

55. Monintainer: an orchestration-independent extensible container-based monitoring solution for large clusters. *Journal of Systems Architecture*. 2023. URL: https://doi.org/10.1016/j.sysarc.2023.103035.

56. Ozdem M. A novel approach for real-time anomaly detection in dynamic computer networks using temporal graph networks and explainable artificial intelligence. *Alexandria Engineering Journal*. 2025. Vol. 132. P. 369–382. URL: https://doi.org/10.1016/j.aej.2025.11.001.

57. Usman M. et al. A survey on observability of distributed edge & container-based microservices. *IEEE Access*. 2022. URL: https://doi.org/10.1109/ACCESS.2022.3193102.

58. Сенів М. М. Методи підвищення доступності та відмовостійкості контейнерної інфраструктури вебпорталів. URL: https://lpnu.ua/sites/default/files/2026/radaphd/35660/dissertationstepanovd.pdf (дата звернення: 21.04.2026).

59. Бочкарьов О. Ю. Автономна децентралізована система моніторингу комп'ютерної мережі на основі програмних агентів. *Computer Systems and Networks*. 2023. Т. 5, № 1. С. 8–19.

60. Burns B., Beda J., Hightower K., Evenson L. Kubernetes: up and running: dive into the future of infrastructure. O'Reilly Media, Inc., 2022.

61. Yin Y., Lin Z., Jin M., Fanti G., Sekar V. Practical GAN-based synthetic IP header trace generation using NetShare. *Proceedings of the ACM SIGCOMM 2022 Conference*. 2022. August. P. 458–472.

62. Soepeno R. A. A. P. Wireshark: an effective tool for network analysis. *CYBV - Introd. Methods Netw. Anal.*. 2023. P. 1–15. URL: https://www.researchgate.net/profile/Ryufath-Soepeno/publication/374675769_Wireshark_An_Effective_Tool_for_Network_Analysis.

63. Huda S., Musthafa M. B., Nogami Y. Zeek intrusion detection on Raspberry Pi for IoT-based agriculture monitoring systems: preliminary investigation. *2024 IEEE International Symposium on Consumer Technology (ISCT)*. IEEE, 2024. August. P. 372–377. URL: https://ieeexplore.ieee.org/abstract/document/10791229/.

64. Sahara A. D., Sapri S., Al Akbar A. The design and implementation of computer network monitoring and security system using Linux Ubuntu Server. *Jurnal Media Computer Science*. 2024. Vol. 3, no. 1. P. 1–16.

65. Alzibdeh N., Alrashdan M. T., Almabhouh A. Bandwidth utilization with network traffic analysis. *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*. IEEE, 2023. December. P. 1–5. URL: https://ieeexplore.ieee.org/abstract/document/10435712/.

66. Colca-Mendoza J., Auccahuasi W. Automated implementation of a TCPDUMP solution for network traffic analysis, strengthening digital security in cyberspace. *2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*. IEEE, 2024. October. P. 1212–1217.

67. Клюшник В. В., Чернецький Є. В., Онищенко О. В. Ідентифікація трафіку мереж передачі даних у реальному часі. *Вісник Приазовського державного технічного університету. Серія: Технічні науки*. 2025. Вип. 50. С. 18–24. URL: https://doi.org/10.31498/2225-6733.50.2025.336234.

68. Method of restoring parameters of information objects in a unified information space based on computer networks / V. Mukhin et al. *International Journal of Computer*

Network and Information Security. 2020. Vol. 12, no. 2. P. 11–21. URL: https://doi.org/10.5815/ijcnis.2020.02.02.

69. Bodenham D. A., Adams N. M. Continuous monitoring of a computer network using multivariate adaptive estimation. *2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW)*, TX, USA, 7–10 December 2013. 2013. URL: https://doi.org/10.1109/icdmw.2013.114.

70. eeDTLS: energy-efficient datagram transport layer security for the Internet of Things / U. Banerjee et al. *2017 IEEE Global Communications Conference (GLOBECOM 2017)*, Singapore, 4–8 December 2017. 2017. URL: https://doi.org/10.1109/glocom.2017.8255053.

71. Eschle J., Gál T., Giordano M., Gras P., Hegner B., Hernandez Acosta U., Kluth S., Ling J., Mato P., Mikhasenko M., Moreno Briceño A., Pivarski J., Samaras-Tsakiris K., Schulz O., Stewart G., Strube J., Vassilev V. Potential of the Julia programming language for high energy physics computing. *Computing and Software for Big Science*. 2023. Vol. 7. URL: https://doi.org/10.1007/s41781-023-00104-x.

72. Postel J. Transmission Control Protocol: DARPA Internet Program Protocol Specification: RFC 793. Internet Engineering Task Force, September 2022. 85 p. URL: https://www.ietf.org/rfc/rfc793.txt (дата звернення: 21.04.2026).

73. Postel J. User Datagram Protocol: RFC 768. Internet Engineering Task Force, August 2017. 3 p. URL: https://www.ietf.org/rfc/rfc768.txt (дата звернення: 21.04.2026).

74. Banerjee U., Juvekar C., Fuller S., Chandrakasan A. eeDTLS: energy-efficient datagram transport layer security for the Internet of Things. *2017 IEEE Global Communications Conference (GLOBECOM 2017)*. 2017. P. 1–6. URL: https://doi.org/10.1109/GLOCOM.2017.8255053.

75. Медзатий Д., Марценюк А. Метод моніторингу параметрів комп'ютерної мережі в реальному часі. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2026. № 2 (травень 2026). С. 49-58.

ДОДАТОК А
(обов’язковий)

Презентація

ЗАДАЧІ ДОСЛІДЖЕННЯ

Поставлена мета досягається розв'язанням таких основних завдань:

- проаналізувати існуючі методи та засоби моніторингу параметрів комп'ютерних мереж, визначити їх переваги та недоліки;
- розробити математичну модель ідентифікації мережевих потоків та формалізувати метрики якості з'єднання у вигляді замкненої системи рівнянь;
- розробити метод пасивного моніторингу на основі eBPF-фільтрів з декомпозицією на підсистеми захоплення, обробки та публікації метрик;
- здійснити програмну реалізацію розробленого методу та провести експериментальне дослідження споживання ресурсів і точності вимірювань в умовах реального мережевого навантаження.

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Наукова новизна отриманих результатів:

- запропоновано метод пасивного моніторингу мережевих потоків, що поєднує фільтрацію заголовків у просторі ядра із статистичною агрегацією метрик у просторі користувача, що дозволяє виконувати діагностику якості з'єднань без копіювання корисного навантаження пакетів та без впливу на мережевий трафік;
- набула подальшого розвитку інформаційна технологія пасивного моніторингу мережевих потоків на основі запропонованого методу.

Практична значимість отриманих результатів полягає у розробленому програмному забезпеченні пасивного моніторингу мережевих потоків eBPF та Python, яке інтегрується з існуючими системами збору метрик Prometheus, Grafana та може бути розгорнуте на пристроях з обмеженими ресурсами, зокрема одноплатних комп'ютерах.

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- Стабільність мережевого з'єднання та виявлення аномалій стають важливими не лише для бізнесу, а і для конкретного приватного користувача, для якого зниження швидкості, збільшення мережевих затримок чи поява аномалій створюють дискомфорт, а також можуть створювати проблеми для професійної діяльності.
- Існуючі системи, очевидно, можуть бути практично застосовувані та виконувати необхідні функції, проте в той же час можуть мати проблемні сторони, бути не надто економними до використання ресурсів або виконувати функції, оперуючи з помірною точністю, цим самим створюючи хибно-позитивні реакції на проаналізовані події.
- Для того, щоб забезпечити оптимальне використання ресурсів і підвищенню ефективності методу і практичній реалізації методу, варто акцентувати увагу на архітектурних рішеннях та оптимізації алгоритму.

БАГАТОРІВНЕВА АРХІТЕКТУРА СИСТЕМИ МОНІТОРИНГУ ГІБРИДНОГО ТИПУ

Архітектура систем моніторингу є досить багатогранною і багаторівневою. Її умовно поділяють на рівень зберігання та представлення результатів, рівень обробки і нормалізації та рівень збирання метрик

АНАЛІЗ ВІДОМИХ МЕТОДІВ

- Недоліки частини готових рішень полягають в тому, що вони не здатні працювати у середовищах, де використання ресурсів та кількість живлення є критичними показниками. Реалізації рішень можуть вимагати значну кількість системних ресурсів, відповідно запускаються лише на промислових серверах, що є недоступним параметром для користувача, який хотів би перевіряти активність та стан власної мережі. При цьому агент сервісу має не заважати самій системі безперебійно оперувати.
- Існуючі рішення досить часто є побудовані на основі сервісів-агентів, що встановлюються на пристрій та можуть активно запускати синтетичні тести, аналізуючи їх результати. Агент відправляє певну кількість пакетів та аналізує кількість отриманих відповідей.

ПОПЕРЕДНЄ МОДЕЛЮВАННЯ МЕТОДУ

- Попереднє моделювання в даному випадку зводиться до етапів, які потрібно виконати під час безпосереднього проєктування:
- 1. Реалізація eBPF програми перехоплення потоків трафіку.
- 2. Створення верифікатора прочитаних заголовків.
- 3. Додавання частини створеного методу для аналізу метрик. Метрики і особливості роботи розробленого методу будуть детально розглянуті у наступному розділі.
- 4. Створення інтерфейсу передачі зібраних подій у користувацький простір.

ПОПЕРЕДНЄ МОДЕЛЮВАННЯ МЕТОДУ

- ❑ 5. Реалізація компонентів простору користувача із окремими модулями, що мають бути ізольовані один від одного.
- ❑ 6. Реалізація модуля, що відповідатиме за вивід результатів в термінал засобами стандартного вводу та виводу, інша частина – Prometheus модуль для уніфікації формату метрик.
- ❑ 7. Інтеграція і тестування в умовах реального навантаження. Поведінка роботи системи в умовах тривалої роботи.
- ❑ 8. Аналіз споживання системних ресурсів, пошук і виправлення можливих втрат пам'яті чи неоптимального використання ресурсів

Попереднє моделювання і логічна структура зводиться до чіткого плану і переліку завдань, який спрямований на те, щоб розподіляти ресурси при розробці реалізації та доопрацюванні методу більш оптимально

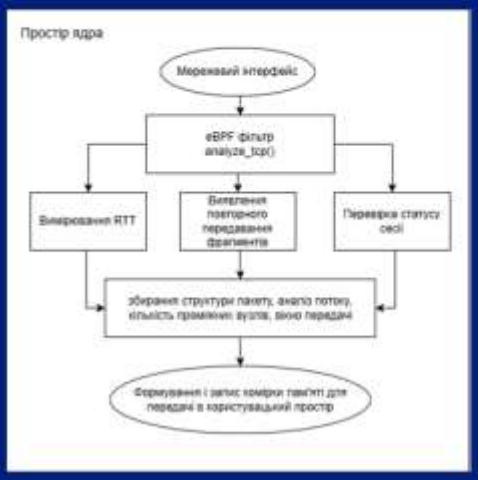
МЕТОД МОНІТОРИНГУ ПАРАМЕТРІВ КОМП'ЮТЕРНОЇ МЕРЕЖІ В РЕАЛЬНОМУ ЧАСІ

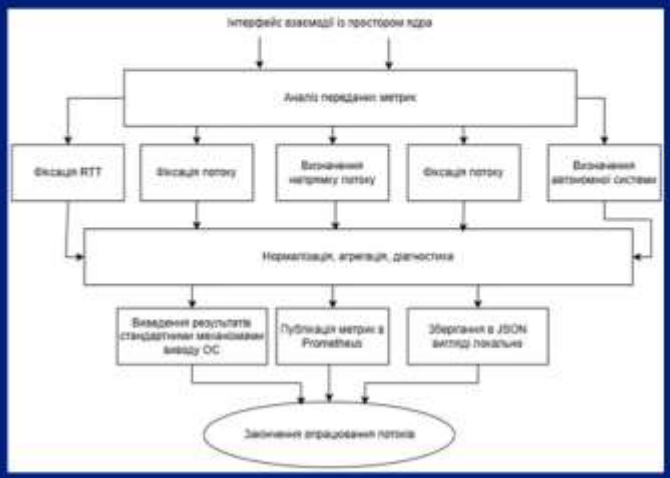
Для точності і можливості широкого спостереження за мережею було визначено наступні метрики, які необхідно визначати і аналізувати:

1. Ідентифікація потоку.
2. Вимірювання RTT (двома методами для більшої точності).
3. Виявлення повторної передачі сегментів.
4. Оцінка кількості проміжних вузлів при передачі даних від джерела до точки призначення.
5. Визначення напрямку трафіку.
6. Обчислення перцентилів і різниці затримок у ковзному вікні.
7. Діагностика першопричини аномалій.

ПОДІЛ МЕТОДУ У ПРОСТОРАХ ОПЕРАЦІЙНОЇ СИСТЕМИ

Блок-схеми роботи методу в частині, що виконується у просторі ядра та користувацькому просторі





РЕАЛІЗАЦІЯ МЕТОДУ

Побудова методу та системи зведена до первинної фільтрації і агрегації метрик у просторі ядра і доопрацювання в просторі користувача. Даний підхід і вирішує проблему управління ресурсами і забезпечує стабільну роботу на пристроях із невеликою кількістю ресурсів, в тому числі і одноплатних комп'ютерів.

Середовище	Файл	Частина логіки
Ядро Linux	bpf/program.c	Збір потоків, виявлення втрат, перевірка стану з'єднань, основна логіка методу
Користувацький простір, Python	monitor.py	Агрегація метрик, діагностика, вивід результатів
Користувацький простір, Python	prometheus.py	HTTP сервер, реєстр метрик та місце де вони публікуються

ЕКСПЕРИМЕНТИ

Споживання ресурсів системою, коли трафіку у мережі мінімальний

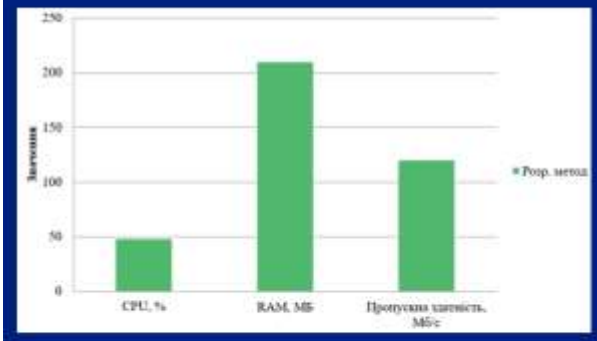
ЕКСПЕРИМЕНТИ

Можливість зниження навантаження на систему шляхом відключення підсистеми експорту метрик Prometheus та модуля запису подій у файл

Після відключення Prometheus було звільнено один мережевий порт та усунуто періодичні HTTP-запити. Було ліквідовано витрати на серіалізацію метрик у текстовий формат, що потребувало обходу всіх внутрішніх структур даних та форматування рядків.
Споживання оперативної пам’яті знизилось на 30-50МБ, навантаження на процесор знизилось на 3-7%. Проведений експеримент показав, що для систем із обмеженими ресурсами, де не потребується довготривале зберігання метрик або інтеграція із зовнішніми системами візуалізації, метод може ще більш оптимально використовувати системні ресурси.

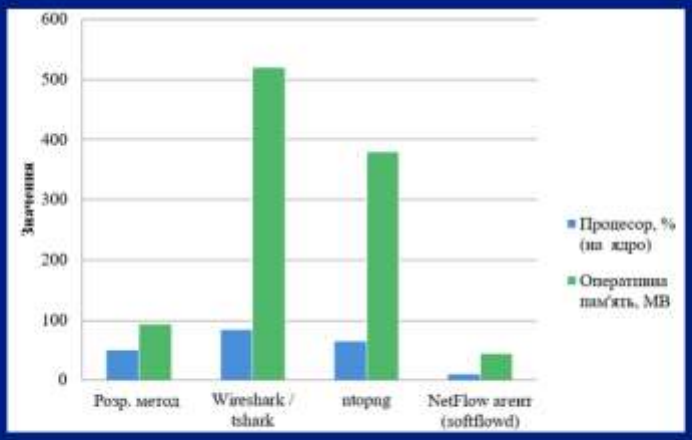
ЕКСПЕРИМЕНТИ

Зростання використання ресурсів відповідно до збільшення мережевого трафіку



ПОРІВНЯЛЬНА СТАТИСТИКА МЕТОДУ

Порівняння споживання ресурсів системи між запропонованим методом та доступними інструментами, спрямованими на моніторинг мережі



ВИСНОВКИ

У кваліфікаційній роботі було розглянуто проблему мережевого моніторингу та виявлення мережевих аномалій, за результатами виконаних теоретичних та практичних досліджень розроблено метод моніторингу параметрів комп'ютерних мереж на основі аналізу мережевих з'єднань, що забезпечує збирання, обробку та візуалізацію метрик якості з'єднання в реальному часі при мінімальному споживанні системних ресурсів в реальному часі.

Впровадження результатів роботи підтверджують, що використання інтелектуальної діагностики аномалій на основі зібраних метрик дозволяє не лише констатувати стан мережі, а й оперативно виявляти причини збоїв, такі як перевантаження буферів чи нестабільність маршрутів. Запропонований підхід забезпечує приріст продуктивності та надійність керування сучасною мережевою інфраструктурою за рахунок пасивного моніторингу параметрів комп'ютерних мереж із використанням технології eBPF, що забезпечує вимірювання затримок (RTT), виявлення повторних передач, оцінку кількості проміжних вузлів та аналіз стану TCP-сесій в реальному часі.

ПУБЛІКАЦІЇ

- Медзатий Д., Марценюк А. Метод моніторингу параметрів комп'ютерної мережі в реальному часі. Вимірювальна та обчислювальна техніка в технологічних процесах. 2026. № 2 (травень 2026).

ДОДАТОК Б
(обов’язковий)

Публікація

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

https://doi.org/10.31891/2219-9365-2026-86-6
УДК 004.9

МЕДЗАТИЙ Дмитро
Хмельницький національний університет
https://orcid.org/0009-0004-3247-6406
e-mail: medzatyi@gmail.com
МАРЦЕНЮК Андрій
Хмельницький національний університет
https://orcid.org/0009-0002-2545-3586
e-mail: andry.martseniyk@gmail.com

МЕТОД МОНІТОРИНГУ ПАРАМЕТРІВ КОМП'ЮТЕРНОЇ МЕРЕЖІ В РЕАЛЬНОМУ ЧАСІ

У даній статті проведено комплексне дослідження та розробку методу моніторингу параметрів комп'ютерної мережі в режимі реального часу, що є критично важливим в умовах глобальної цифровізації та зростання складності мережевих архітектур. Автором обґрунтовано, що традиційні підходи, засновані на періодичному зборі статистичних даних, не здатні забезпечити необхідну швидкість реакції на динамічні зміни трафіку та сучасні кіберзагрози. Особливу увагу приділено аналізу недоліків існуючих систем, які використовують повне копіювання пакетів із простору ядра в простір користувача, що призводить до надмірного споживання ресурсів процесора та пам'яті. Наукова новизна роботи полягає у запропонуванні багаторівневої моделі вимірювання характеристик TCP-з'єднань на основі технології eBPF (extended Berkeley Packet Filter). Цей підхід дозволяє здійснювати фільтрацію та первинну агрегацію метрик безпосередньо у просторі ядра операційної системи, мінімізуючи накладні витрати на копіювання даних. Розроблений метод забезпечує точне вимірювання затримки (RTT) двома способами: шляхом аналізу фаз встановлення сесії (SYN/SYN-ACK) та через обробку опцій TCP Timestamp. Математично формалізовано процеси ідентифікації мережевих потоків, визначення напрямку трафіку, оцінки кількості проміжних вузлів через TTL та обчислення рівня втрат пакетів. Програмна реалізація методу базується на комбінованому використанні мов C та Python, де низькорівнева обробка виконується в ядрі, а високорівневий аналіз, діагностика аномалій та інтеграція з системами збору метрик (Prometheus) – у користувацькому просторі. Окремо висвітлено механізми виявлення ретрансмісій та класифікації мережевих аномалій за допомогою набору діагностичних правил. Запропоноване рішення демонструє високу ефективність на пристроях із обмеженими апаратними ресурсами, зокрема на одноплатних комп'ютерах, забезпечуючи стабільність моніторингу навіть за умов високої інтенсивності трафіку.

Ключові слова: моніторинг мережі, реальний час, eBPF, TCP/IP, RTT, простір ядра, простір користувача, мережеві аномалії, ретрансмісія, QoS.

MEDZATYI Dmytro, MARTSENIUK Andrii
Khmelnytskyi National University

METHOD OF MONITORING COMPUTER NETWORK PARAMETERS IN REAL TIME

This article presents a comprehensive study and development of a method for monitoring computer network parameters in real time, which is critically important in the context of global digitalization and the increasing complexity of network architectures. The author substantiates that traditional approaches based on periodic collection of statistical data are unable to provide the necessary speed of response to dynamic traffic changes and modern cyber threats. Particular attention is paid to the analysis of the shortcomings of existing systems that use full copying of packets from kernel space to user space, which leads to excessive consumption of processor and memory resources. The scientific novelty of the work lies in proposing a multi-level model for measuring TCP connection characteristics based on eBPF (extended Berkeley Packet Filter) technology. This approach allows filtering and primary aggregation of metrics directly in the kernel space of the operating system, minimizing the overhead of data copying. The developed method provides accurate measurement of delay (RTT) in two ways: by analyzing the session establishment phases (SYN/SYN-ACK) and by processing TCP Timestamp options. The processes of identifying network flows, determining the direction of traffic, estimating the number of intermediate nodes via TTL, and calculating the packet loss rate are mathematically formalized. The software implementation of the method is based on the combined use of the C and Python languages, where low-level processing is performed in the kernel, and high-level analysis, anomaly diagnostics, and integration with metrics collection systems (Prometheus) are performed in user space. The mechanisms of detecting retransmissions and classifying network anomalies using a set of diagnostic rules are separately highlighted. The proposed solution demonstrates high efficiency on devices with limited hardware resources, in particular on single-board computers, ensuring stable monitoring even under conditions of high traffic intensity.

Keywords: network monitoring, real-time, eBPF, TCP/IP, RTT, kernel space, user space, network anomalies, retransmission, QoS.

Стаття надійшла до редакції / Received 09.01.2026
Прийнята до друку / Accepted 02.03.2026
Опубліковано / Published 14.05.2026

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Актуальність розробки та впровадження методів моніторингу параметрів комп'ютерної мережі в реальному часі зумовлена стрімкою цифровізацією всіх сфер людської діяльності, де мережева

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

інфраструктура виступає фундаментальним базисом для обміну даними. У сучасних умовах навіть короткочасна затримка або збій у роботі мережі можуть призвести до критичних фінансових втрат, репутаційних ризиків та зупинки стратегічно важливих бізнес-процесів. Традиційні методи аналізу, що базуються на періодичному зборі статистичних даних, вже не здатні забезпечити необхідну швидкість реакції на динамічні зміни в трафіку, що вимагає переходу до безперервного спостереження [1].

Зростання складності мережевих архітектур, включаючи гібридні хмарні середовища, програмно-конфігуровані мережі (SDN) та інтернет речей (IoT), створює нові виклики для систем адміністрування. Велика кількість різнорідних пристроїв генерує колосальні обсяги даних, які необхідно обробляти миттєво для підтримки стабільності з'єднань. Моніторинг у реальному часі дозволяє виявляти вузькі місця в інфраструктурі ще до того, як вони стануть причиною повної відмови системи, що забезпечує високу доступність сервісів для кінцевих користувачів [2].

Одним із ключових аспектів актуальності є питання кібербезпеки, оскільки сучасні загрози, такі як DDoS-атаки або несанкціоноване проникнення, розвиваються надзвичайно швидко. Тільки інструменти моніторингу в реальному часі здатні зафіксувати аномальні сплески активності або підозрілу поведінку пакетів даних у момент їх виникнення. Це дає можливість автоматизованим системам захисту негайно ізолювати загрозу, мінімізуючи потенційні збитки від витоку конфіденційної інформації чи пошкодження цілісності даних [3].

Розвиток мультимедійних технологій, зокрема потокового відео високої чіткості, IP-телефонії та онлайн-конференцій, висуває жорсткі вимоги до якості обслуговування (QoS). Параметри затримки (latency), джитера (jitter) та втрати пакетів мають критичне значення для комфортної роботи користувача. Постійний контроль цих показників дозволяє динамічно перерозподіляти мережеві ресурси та оптимізувати маршрутизацію трафіку в реальному часі, адаптуючись до поточного навантаження на канали зв'язку [4].

Впровадження концепції Індустрії 4.0 та автоматизація виробничих процесів потребують гарантованої надійності мережевих з'єднань для координації роботи промислових роботів та датчиків. У таких системах часові межі реакції вимірюються мілісекундами, тому будь-який метод моніторингу, що працює з запізненням, є неприйнятним. Актуальність теми підтверджується необхідністю створення інтелектуальних систем, які інтегрують алгоритми машинного навчання для прогнозування стану мережі на основі поточних параметрів [5].

Економічна ефективність експлуатації мереж також напряму залежить від якості моніторингу, оскільки проактивне обслуговування обходиться значно дешевше, ніж аварійне відновлення після катастрофічних збоїв. Можливість бачити повну картину розподілу навантаження в реальному часі дозволяє компаніям більш раціонально планувати масштабування інфраструктури та уникати витрат на надлишкові потужності. Це робить розробку нових методів моніторингу пріоритетним завданням для IT-департаментів великих корпорацій та провайдерів телекомунікаційних послуг [6].

Перехід на віддалений формат роботи по всьому світу значно збільшив навантаження на корпоративні VPN-шлюзи та зовнішні вузли доступу. Забезпечення стабільного та безпечного підключення для тисяч співробітників, які перебувають у різних географічних локаціях, вимагає від адміністраторів інструментів для миттєвої діагностики проблем на будь-якій ділянці ланцюга передачі даних. Моніторинг у реальному часі стає "очима" системного адміністратора, дозволяючи бачити стан віддалених сегментів мережі без затримок [7].

Крім того, сучасні стандарти передачі даних та протоколи стають дедалі складнішими, що ускладнює процес ручного пошуку несправностей. Автоматизовані методи збору та візуалізації параметрів мережі в режимі реального часу допомагають знизити навантаження на технічний персонал, зменшуючи ймовірність людської помилки. Це актуально в контексті загального дефіциту висококваліфікованих кадрів, оскільки дозволяє автоматизувати рутинні операції з перевірки цілісності мережі [8].

З появою технологій 5G та розвитком «Розумних міст» обсяги трафіку продовжують зростати в геометричній прогресії, що робить старі підходи до моніторингу остаточно застарілими. Нові методи мають враховувати специфіку високошвидкісних каналів та здатність обробляти великі масиви даних (Big Data) безпосередньо "на льоту". Науковий пошук у цьому напрямку є критично важливим для створення стійкої цифрової екосистеми майбутнього, де мережа є не просто допоміжним інструментом, а життєво важливою артерією суспільства.

Отже, актуальність дослідження методів моніторингу параметрів комп'ютерної мережі в реальному часі зумовлена сукупністю технологічних, безпекових та економічних чинників. Необхідність забезпечення безперервності бізнесу, захисту від кібератак, підтримки високої якості сервісів та підготовки інфраструктури до викликів майбутнього робить цю тему однією з найважливіших у сучасній комп'ютерній інженерії. Розробка ефективного методу дозволить не лише констатувати стан мережі, а й активно керувати її продуктивністю, гарантуючи надійність у будь-яких умовах експлуатації.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Принцип роботи моніторингу параметрів мережі

В цілому мережевий моніторинг із розвитком IT-інфраструктури став постійною складовою керування й адміністрування інфраструктури в цілому. Загальне поняття моніторингу враховує аспекти періодичного або постійного збирання, аналізу, обробки метрик про стан взаємодії пристроїв, сервісів із ресурсами або пристроями, що розташовані із ними у одній і тій же локальній мережі, або що розташовані віддалено із доступом через випадкові проміжні вузли глобальних мереж. Поняття фіксованого рівня обслуговування(QoS) порівняно не часто використовується для типових користувацьких мереж та мереж дрібного бізнесу, але для мереж із великими кількостями трафіку поняття є досить актуальним.

Стандарт ISO/IEC 27033 переважно орієнтується на аспекти безпеки при моніторингу, проте ефективна система моніторингу має також забезпечувати опрацювання подій в реальному часі, журналювання, збереження метрик та засоби сповіщення адміністратора.

Засоби, спрямовані на мережеву телеметрію, можна поділити на категорії, класифікуючи їх за способом збирання метрик. Активні і пасивні засоби напряму відрізняються способами.

Активні методи зазвичай генерують тестовий трафік, окремі потоки, на основі яких можна виміряти параметри мережевого з'єднання на шляху до точки призначення. Активні засоби створюють синтетичний трафік.

Досить популярними є засоби для ICMP-тестування, використовуючи утиліти ping, traceroute(частина реалізацій утиліти застосовує саме ICMP). Для визначення маршруту застосовуються traceroute і mtr. Щоб виміряти пропускну здатність часто застосовують iperf3. Окремо можна визначити засоби тестування, що починають свою роботу із прикладного рівня, тим не менш в кінцевому результаті тести прикладного рівня все одно відображають більш низькорівневе тестування на 2-3-4 рівнях моделі OSI.

Активний метод аналізу з'єднання має ряд переваг і недоліків. Щодо переваг, то тестування можна провести у довільний час, адаптуючи вхідні параметри так, щоб результати були найбільш показовими. Водночас недоліком є те, що трафік є повністю синтетичним і за певних умов не покаже проблему, яка існує для справжнього трафіку системи. Таке вимірювання характеризує лише поведінку синтетичного трафіку. Окрім цього, генерування окремих потоків – це додатковий трафік, що циркулює мережею, великі об'єми трафіку при тестах iperf3 можуть завантажити канал і напряму негативно впливати на справжній трафік користувача. Також синтетичний трафік може піддаватись блокуванню від мережевих екранів і засобів вторгнення.

Пасивні засоби спрямовані на аналіз існуючого трафіку, не генеруючи нового. Глибокий аналіз заголовків пакетів можуть дати розуміння процесів, що відбуваються в мережі на основі пакетів, що прибули до пристрою. Аналіз концентруєтеся на трафікові застосунків і сервісів. Пасивні засоби не впливають на канал зв'язку та не спотворюють вибірку потоків синтетичними даними. Вплив мережевих екранів на аналізу також не притаманний для цього способу. Трафік, що циркулює від пристрою, до сервісу, який розташований на сервері має бути дозволеним. Якщо сервер не готовий приймати запити, у випадку TCP ми не побачимо встановленої сесії, у випадку UDP/ICMP – не побачимо відповідей.

Окремо можна виділити гібридні засоби. Гібридні засоби моніторингу можуть застосовувати і активні, і пасивні тестування мережі. Таким чином, коли пасивний аналіз зафіксував аномалію – активний аналіз може допомогти локалізувати проблему, не створюючи надмірне навантаження.

Підвищення рівнів складності в рішеннях мережевої інфраструктури відповідно адаптує задачу моніторингу для контролю за мережевими з'єднаннями і якістю передачі даних. Більш традиційні підходи, що використовують як основні метрики – метрики операційної системи, а саме завантаження каналу мережі, завантаження процесора, кількість вільної оперативної пам'яті, проявляють недолік у вигляді статичності. Особливість полягає в тому, що стан каналу і мережі в цілому – для різних потоків даних може бути різний. Це класична особливість підходу комутації пакетів при передачі даних із точки А в точку Б.

Об'єктом дослідження систем моніторингу є процес передачі даних у мережах TCP/IP на рівні окремих мережевих потоків. Мережевий потік - це послідовність пакетів, у випадку TCP, окремі сегменти можна корелювати в єдиний потік за допомогою номера послідовності, IP адреси джерела та призначення та портами джерела і призначення.

Розглянемо завдання систем моніторингу із векторної точки зору. Нехай $F=\{f1,f2,\ldots,fn\}$ - множина TCP-потоків, де кожен потік *fn* визначається чотирма параметрами IP адреси ініціатора, IP призначення, мережевий порт ініціатора та порт призначення *(src_ip, dst_ip, src_port, dst_port)*.

Для кожного *fn* можна визначити вектор характеристик:

$$Q(fn) = \{RTT,\ \sigma RTT,\ L,\ W,\ H\}$$

де RTT - затримка, σRTT - джитер, L - рівень втрат пакетів, W - розмір вікна прийому, H - кількість проміжних вузлів.

Принцип обробки трафіку системами моніторингу, що базуються на пасивному аналізі

Оскільки моніторинг цього типу базується на спостереженні за справжнім трафіком, що передається по каналу зв'язку, то система переважно працює як окремий сегмент отримувача.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

Для аналізу можуть застосовуватись або повні копії кадрів, пакетів, сегментів і датаграм, або лише їх заголовки із ключовими полями. На відмінну від систем протилежного типу, де аналізуються синтетичні дані, в системі сегменті-отримувача з'являється ускладнення – є потреба аналізувати потік трафіку в реальному часі, не маючи можливості виконати повторний запит. Іншою проблемою, що потребує вирішення, є обмежений контекст. Проаналізувати TCP сесію, без аналізу етапу встановлення і трьох етапної комунікації досить складно.

Система отримує потік фрагментів, що не відносяться один до одного, в довільній послідовності. Якщо аналізувати саме транспортний рівень моделі OSI сегменти, датаграми несуть лише поля заголовку: порядковий номер, розмір вікна прийому, часові мітки, у випадку TCP – стани сесії. Жодне із цих полів не дозволяють одразу зробити висновок про стан з'єднання. Відповідно система, що аналізує справжні пакети є автоматом відстеження стану з'єднань та передбачає механізм поєднання пакетів за ознаками, щоб встановити їх спорідненість. Вигляд формату TCP заголовка подано на рис. 1.

Переважна більшість існуючих методів застосовують повне копіювання пакетів із простору ядра операційної системи в простір користувача із корисним навантаженням. Пакет потрапляє через мережевий інтерфейс в буфер, інструменти типу Wireshark, Zeek досить часто застосовують цей спосіб. Спосіб має чимало готових до використання бібліотек та є досить універсальним в розрізі використанні для систем моніторингу. Проте має значні недоліки, одним із яких є надмірна надлишковість. Для того, щоб оцінити стан мережі, системі моніторингу не потрібно аналізувати поля із інформацією (поле data в заголовку). Іншим недоліком є копіювання пакету в користувацький простір, при рості використання пропускної здатності, система аналізу із невеликою кількість ресурсів не зможе коректно працювати через виснаження ресурсів процесора, часу використання ядер і оперативної пам'яті. Також при надсиланні надмірної кількості трафіку і виснаження ресурсів, операційна система за стандартним механізмом роботи може розпочати відкидання пакетів. Таким чином, результати вимірів можуть бути спотворені.

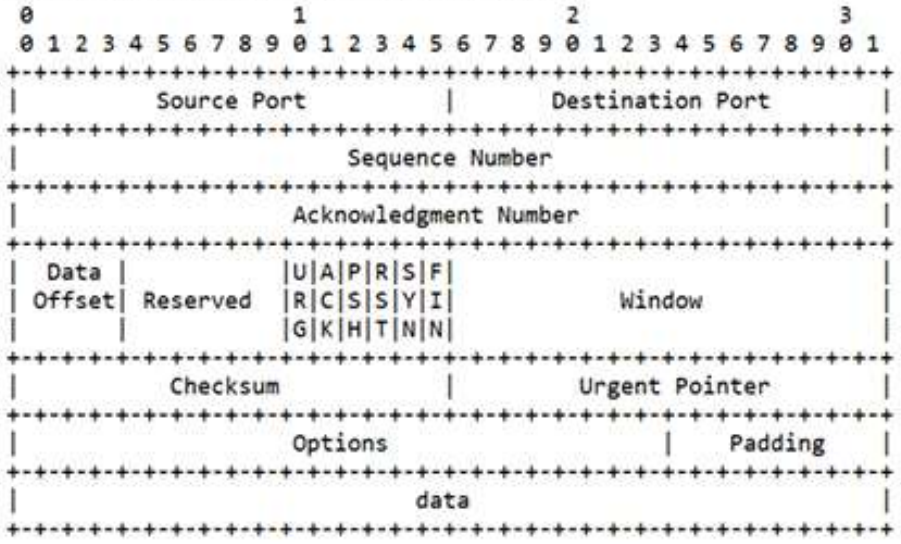


Рис. 1. Вигляд формату TCP заголовка [9]

Можливе також застосування технології TAP (Test Access Point), це фізичний пристрій, що дублює трафік на фізичний інтерфейс, звідки трафік можна зняти для подальшої обробки. Це апаратний пристрій, який здатен захоплювати і передавати кадри із високим рівнем точності. Головні недоліки - фізичне втручання в інфраструктуру, встановлення в кожному сегменті мережі, складність масштабування і створення додаткової точки відмови. Ця технологія чудово підходить для аналізу мережевих загроз та засобів виявлення вторгнень, однак надмірна надлишковість не дасть можливості застосувати технологію із пристроями, що мають мінімальну кількість ресурсів.

Із існуючих способів також існують протоколи типу NetFlow, протоколи агрегованої телеметрії. За допомогою цього методу можна отримати одразу агреговані потоки за класифікаціями, водночас загальна статистика не є надто детальною і її не є достатньо для аналізу параметрів і стану мережі. За допомогою стандартного NetFlow важко визначити аномалії канального і мережевого рівнів.

Порівняно недооціненим методом є захопления і фільтрація заголовків за допомогою фільтрів eBPF (extended Berkeley Packet Filter). Це код, що виконується напряму у просторі ядра без переведення пакетів в користувацький простір для аналізу. Аналіз трафіку може відбуватись без виходу за межі простору ядра, а в користувацький простір повертаються лише дані, які були відфільтровані. Розмір пакету зменшується в декілька десятків разів, можливість обробки корисних даних збільшується відповідно. Основні складнощі при

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

створенні реалізацій - це чітко визначений синтаксис, глибокий аналіз логіки, верифікація перед запуском, щоб уникнути неочікуваних умов в коді. Це складність, яка з'являється для реалізацій, але однозначно не недолік, це засіб захисту системи і спосіб забезпечити стабільну роботу операційної системи, недопустивши неочікувані умови до виконання.

Обмеження eBPF полягають в тому, що існує залежність від версії ядра Linux 4.9 (дата офіційного виходу грудень 2016 року), на більш старіших версіях були впровадженні певні функції eBPF, але частина із функцій додавались із оновленнями до версії 4.9.

Таким чином, незалежно від застосованих технологій та специфік тих чи інших реалізацій, основне завдання у випадку відстеження TCP сесій зводиться до відновлення стану TCP-сесій та розбір і пошук спорідненості датаграм UDP із великого потоку даних. Протокол TCP є з'єднано орієнтованим і збір метрик має місце в контексті конкретної сесії. Система може почати працювати із вже встановленими з'єднаннями, без видимості трьох етапного рукостискання(handshake), відповідно такі сесії треба обробляти окремо за іншим механізмом, щоб не створювати хибних метрик.

Глобальні мережі не є ідеальним середовищем із ідеальною точністю передачі даних, відповідно прибування пакетів може відбуватись не в початковій послідовності, а в перевпорядкованому форматі. Це прийнятна поведінка мережі, однак і рішення моніторингу потрібно адаптовувати відповідно.

Також система має спостерігати за сигналами FIN та RST і проактивно очищати записи про завершення сесій.

Попередньо зазначались особливості опрацювання саме TCP потоків трафіку. Однак існує не менш важливий транспортний протокол UDP, що працює без встановлення з'єднання і не має механізму підтвердження доставки датаграми.

Через специфіку роботи протоколу моніторинг має бути побудований із іншим підходом для збирання телеметрії, в цілому моніторинг є більше обмежений в кількості корисних даних, на основі яких можна зробити висновки про роботу мережі.

Кількість UDP датаграм в мережах та їх функції є значними. Велика кількість протоколів використовують саме UDP на транспортному рівні, щоб зменшити затримки при передачі даних за рахунок уникання повторного надсилання даних та відсутності механізму підтвердження. Вигляд заголовку UDP подано на рис 2.

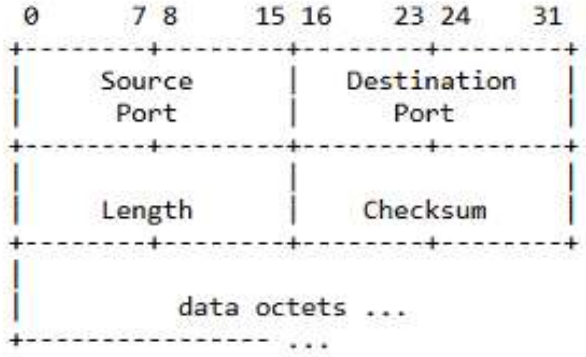


Рис. 2. Вигляд заголовка UDP протоколу [10]

Для UDP датаграм можна також визначити корисну інформацію – це інтенсивність потоку із пропускною здатністю, міжпакетний інтервал. Та однією із найбільш цінних метрик є рівень втрат і зміна послідовність датаграм, що визначена із порядкового номера пакету.

Вимірювання RTT значень для UDP є досить нестандартним завданням, враховуючи відсутність стандартного механізму відповіді на транспортному рівні.

У випадках аналізу трафіку протоколів DNS, NTP, RADIUS на прикладному рівні можна отримати більше корисних деталей, приміром ідентифікатор транзакції.

Використання комбінованих даних із транспортного рівня в парі із даними зібраними на прикладному рівні дає більш детальний результат. Таким чином, можна пов'язати запит-відповідь до транзакції та оцінити стан мережі і проміжних пристроїв через які пройшов UDP трафік до точки призначення.

Значного поширення набув протокол DTLS, призначений для шифрування датаграм. Шифрування датаграми не дозволяє проаналізувати вкладення вище мережевого рівня. Лише доступний заголовок IP рівня, адреси, порти, розмір пакетів і часові мітки. Набір інформації є досить обмеженим, однак все ще можна визначити пропускну здатність і різницю затримок між пакетами.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

У випадку DTLS завдання повністю зводиться до класифікації застосунків і пошуку аномалій без можливості аналізу і розшифрування вмісту датаграми і даних енкапсульованих на вищому рівні. Існує декілька версій DTLS, різниця між DTLS 1.2 і DTLS 1.3 подана на рис. 3.

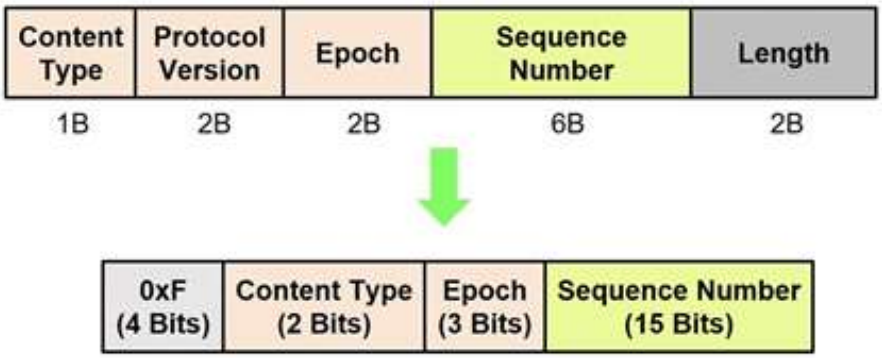


Рис. 3. Вигляд заголовки DTLS протоколу та різниця між версіями DTLS 1.2 і 1.3 [11]

Метод моніторингу параметрів комп'ютерної мережі в реальному часі

Враховуючи результати попереднього аналізу, було вирішено будувати рішення багаторівневої моделі вимірювання характеристик TCP з'єднань на основі аналізу заголовків пакетів у просторі ядра операційної системи засобами eBPF. Основним методом є захоплення і фільтрація заголовків за допомогою фільтрів eBPF. Метод є досить оптимальним для одноплатних комп'ютерів, оскільки керування ресурсами системи здійснюється із порівняно меншими втратами.

Для точності і можливості широкого спостереження за мережею було визначено наступні метрики які необхідно визначати і аналізувати:

1. Ідентифікація потоку.
2. Вимірювання RTT (двома методами для більшої точності).
3. Виявлення повторної передачі сегментів.
4. Оцінка кількості проміжних вузлів при передачі даних від джерела до точки призначення.
5. Визначення напрямку трафіку.
6. Обчислення перцентилів і джитеру у ковзному вікні.
7. Діагностика першопричини аномалій.

В межах етапу проєктування варто формалізувати вимірювання параметрів та подання метрик у математичній формалізації.

Ідентифікація мережевого потоку відбувається за умови, що пакет пройшов через конкретний мережевий інтерфейс та побудувавши ключ на основі набору даних у полях пакету:

$$k = (src\_ip, dst\_ip, sport, dport) \in \mathbb{N}^2 \times \{0..65535\}^2$$

Визначити напрямок пакетів можливо, перевіривши чи належить IP адреса ініціатора трафіку множині IP адрес, налаштованих на локальному інтерфейсі L:

$$dir(k) = \{\, 1\ (OUT), \text{якщо } src\_ip \in L;\ \ 0\ (IN), \text{якщо } src\_ip \notin L \,\}$$

Вимірювання часу RTT можна двома способами. Один із способів - це орієнтування на SYN/SYN-ACK стани TCP сесії, який можна використовувати як резервний, коли мережеве обладнання не підтримує опції TCP Timestamp. При встановлені сесії відправник ініціює встановлення сесії фрагментом SYN, прапорець SYN еквівалентний 1, ACK еквівалентний 0. Фільтр eBPF має зафіксувати проходження фрагменту через інтерфейс і фіксує часову мітку ядра:

$$entry\_timestamp[k] \leftarrow T\_syn = bpf\_ktime\_get\_ns()$$

При надходженні відповіді SYN-ACK сервіс моніторингу має здійснити пошук за оберненим ключем k і здійснює розрахунок RTT як різницю часових міток між фрагментами, один із яких був ініціацією, а інший відповіддю на ініціацію:

$$RTT^1 = T_{synack} - T\_syn,\ \ T\_synack = bpf\_ktime\_get\_ns()$$

Альтернативним засобом виміру є метод, що аналізує TCP Timestamp. Згідно із RFC 7323, якщо обидві сторони підтримують опцію TSOPT, то в TCP заголовку як додаткові поля будуть доступні TSval і TSecr, тобто лічильник відправника і значення, що відповідає за останній отриманий пакет. В реалізації фільтрування і аналізу опцій за допомогою eBPF відбувається ітеративний розбір опцій:

$$RTT_2 = T\_ack - sent\_ns,\ \text{ якщо } tsopt\_map[k].tsval = TSecr(\text{пакету})$$

Значення RTT може бути коректним за умови:

$$50000 \le RTT_2 \le 5000000000\ \text{(нс)}$$

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

В умові достовірності верхня і нижня межа підібрані експериментально. Нижня межа потрібна, щоб не враховувати хибні спрацювання на фрагменти, що прибули в невпорядкованому форматі. Верхня межа спрямована на те, щоб не враховувати застрілі сесії, які не продовжуються впродовж 5 секунд і, можливо, не є закритими коректно.

Основний спосіб виявлення умов перенадсилання пакетів(ретрансмісій) – це пошук співпадінь порядкових номерів фрагментів, що відносяться до одного і того ж потоку із різницею в часовому проміжку визначеного порогу:

$$is\_ret(p) = 1, \text{ якщо } last\_seq[k].seq = SEQ(p) \wedge \Delta T > T\_ret$$

Значення T_ret визначено як 50мс. Це нижня межа часу очікування повторної передачі фрагменту. Це значення дозволяє відрізнити повторну передачу від перевпорядкованих фрагментів.

Оцінку проміжних вузлів можна реалізувати за допомогою дослідження і варіації значень TTL. В залежності від пристрою стандартні початкові значення можуть відрізнятись, приміром Android та Linux мають стандартне значення TTL 64, операційні системи Windows та MacOS має початкове значення 128, також в залежності від реалізацій мережевого стеку операційних систем це значення може бути іншим.

$$H = TTL_0 - TTL(p)$$

Визначається як різниця стандартного значення із тим, яке потрапило у фільтр. Значні стрибки кількості проміжних вузлів на одному і тому ж потокові трафіку можуть свідчити про глобальні зміни маршрутизації.

Існують випадки, коли проміжні пристрої можуть двічі віднімати значення TTL при передачі фрагменту далі, тому різниця у одну-дві одиниці від середнього значення можна вважати як похибку або балансування трафіку на проміжних вузлах.

Щодо оцінювання рівня втрачених фрагментів за інтервал часу, то за одиницю часу спостереження система фіксує кількість повторних передач і загальну кількість пакетів для пристрою, що веде комунікацію. Це співвідношення кількості повторних передач до загальної кількості передач. Рівень втрат можна подати у наступному вигляді:

$$L(\Delta t) = N\_ret \,/\, (N\_total + N\_ret)$$

В контексті втрат також можна визначити рівень порогу, після перетину якого система почне сигналізувати адміністратора.

Розглянувши попередньо опції, які можуть бути використанні для програмної реалізації, оцінивши ризики, було прийнято рішення застосовувати комбіноване рішення із мов програмування C та Python.

Мова програмування C є необхідною для взаємодії саме із ядром і фільтром eBPF, це забезпечує низькорівневу обробку трафіку, без переміщень пакетів у користувацький простір. Швидкодія мови програмування C є перевагою для загальної швидкодії системи.

Мова програмування Python в контексті побудови програмної реалізації вирішує інше завдання. Створення програмних інтерфейсів для міжсервісної взаємодії, обробки і аналізу тих результатів, які були отримані від eBPF, основна функція, на яку планується використати мову програмування Python. Акцент на швидкодії робиться саме у тому місці, де можливе надвисоке навантаження через сплески трафіку.

Технологія eBPF є не зовсім ідентичною до класичних підходів програмування у користувацькому просторі. Так як виконання відбувається у просторі ядра, відповідно синтаксис і доступні конструкції є обмеженими. Обмеження пов'язанні із роботою верифікатора ядра перед безпосереднім завантаженням коду.

Статичний верифікатор ядра перевіряє кожну процедуру і конструкцію до виконання. Верифікатор будує граф потоку управління і симулює виконання всіх шляхів, аналізуючи зміни регістрів і поведінку програми. Будь - яке відхилення від дозволених конструкцій спричинить виведення діагностичного повідомлення і виконання буде неможливим до ре-конфігурації синтаксису до норми. Навіть за умови успішної компіляції код може бути недоступним до фактичного виконання.

Таким чином, при розробці алгоритмів потрібно окремо від вимог компілятора акцентувати увагу і на вимоги верифікатора. Суттєвим і помітним обмеженням є заборона на використання циклів без статичної термінації і чітко визначеної кількості ітерацій виконання. Приміром цикл із break не може бути верифікованим, навіть якщо ми гарантовано впевнені в існуванні умови, при якій цикл закінчиться.

При створенні eBPF програм потрібно передбачити механізм передачі станів. BPF мапа - це структура, що існує в просторі ядра, але водночас доступна із простору користувача. Для хеш-таблиць BPF_HASH доступні декілька операцій - пошук, оновлення і видалення. Особливістю у порівнянні із хеш-таблицями є те, що BPF_HASH таблиці мають статичний розмір, під який виділяється пам'ять у момент запуску. Перелічену специфіку не можна позиціонувати як недолік, це правила, яких варто дотримуватись і враховувати при написанні коду.

У методі, що розробляється ми намагатиемось не переносити надлишкові дані між просторами ядра і користувача, щоб зберегти ресурси системи, однак повного перенесення уникнути не вдасться. Технічно передача даних між просторами можлива завдяки окремим вказівникам запису на рівні ядра і читання на рівні користувача. Тобто, якщо ми хочемо перенести значимі поля пакету розміру 40-80 байт замість повних 1500

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

байт, то на рівні ядра ми записуємо дані в буфер, виконуючи функцію perf_submit, а для зчитування в просторі користувача виконується функція perf_buffer_pool.

Через надмірну кількість трафіку і високу інтенсивність, яку операційна система не може опрацювати через брак ресурсів, відбувається переповнення буферу, яке може здійснити вплив на загальну статистику.

Для точного обчислення інтервалів часу оптимальним варіантом є використання часу ядра у наносекундах замість сторонніх залежностей. Опитати час ядра можна завдяки функції bpf_ktime_get_ns, із точність до 1нс. Блок-схема алгоритму частини рішення, що виконується у просторі ядра, зображено на рис. 4.

Поділ виконання частин методу на два окремих простори також спрямований на підвищення безпеки системи. Взаємодія між просторами відбувається у формі однонаправленого потоку. З міркувань практичності і безпеки поділ реалізований таким чином, що у просторі ядра відбувається доступ і фільтрація потоків трафіку, а у просторі користувача виконуються частини, для яких необхідна динамічна пам'ять. В користувацькому просторі опрацьовуються складні структури даних, доступна підтримка більшої кількості бібліотек мов програмування. Також у користувацькому просторі більш доцільно проводити агрегацію статистик.

Як інструмент взаємодії між мовами програмування Python і C застосована бібліотека BCC, під час виконання частини Python коду, відбувається компіляція C коду. Такий підхід дозволяє вносити зміни в C код і тестувати результати одразу на пристрої, без необхідності використання бінарних файлів. Для застосування бібліотеки необхідно передавати метадані ядра.

Відлагодження eBPF застосунків є значно складнішим порівняно із звичайним C кодом, для тестування і фіксації виведень застосовується трасування ядра за шляхом /sys/kernel/debug/tracing/trace_pipe. Тобто саме для проведення тестувань скриптів немає можливості виведення результатів відладжування в термінал через засоби стандартного виводу, результати можна побачити після виконання, переглянувши вміст файлу.

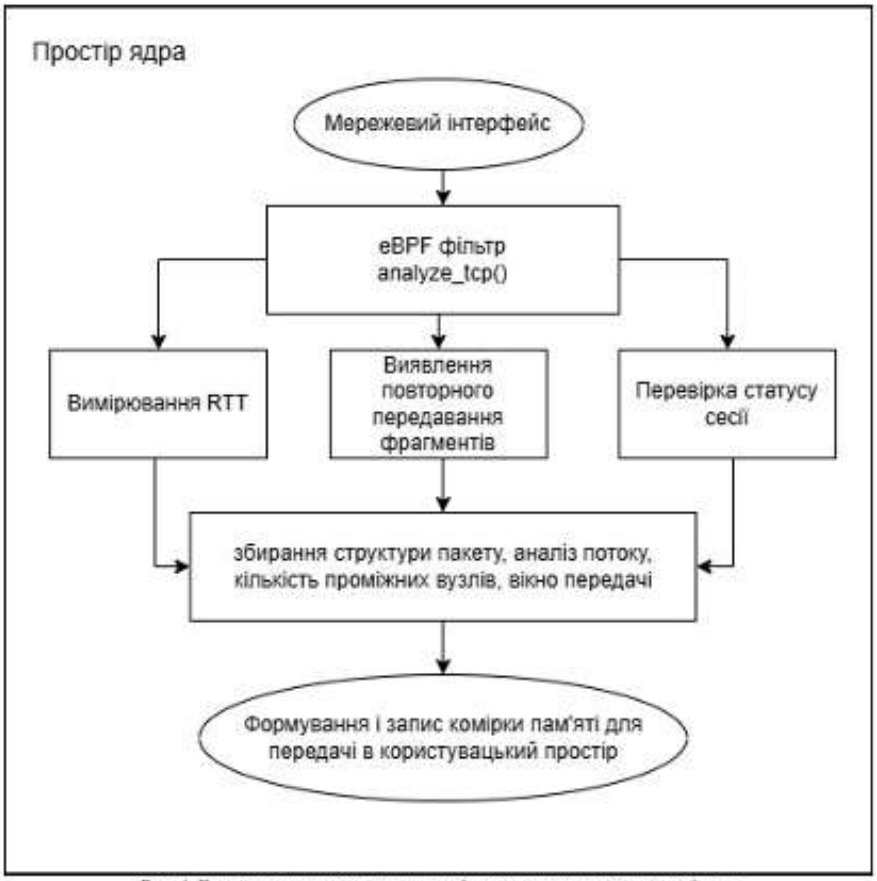


Рис. 4. Блок-схеми алгоритму в частині, що виконується у просторі ядра

Таким чином, побудова методу та системи зведена до первинної фільтрації і агрегації метрик у просторі ядра і доопрацювання в просторі користувача. Даний підхід і вирішує проблему управління

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

ресурсами і забезпечує стабільну роботу на пристроях із невеликою кількістю ресурсів, в тому числі і одноплатних комп'ютерів.

У порівнянні із існуючими системами доступними на ринку, саме цей підхід демонструє приріст у продуктивності та зменшення використання системних ресурсів через вдосконалений засіб обробки потоків трафіку.

Якщо більш детально розглянути іншу частину реалізації методу у просторі користувача, то вона створена на Python3. Під час виконання ця частина запускається як єдиний процес, що взаємодіє із частиною у просторі ядра.

Протягом всього часу виконання, відбувається вичитування записів із буферу ядра через інтервал у 100мс. Функція виконується у однопоточному режимі, механізм псевдо багатопоточності у скрипті застосовується для DNS запитів та сервера метрик Prometheus.

Як було зазначено раніше, на етапі виконання вивчаються локальні IP адреси на основі активних мережевих інтерфейсів, щоб спростити процес ідентифікації пакетів на вхідні і вихідні. Саме у цьому просторі відбувається безпосереднє обчислення і визначення параметрів на основі отриманих даних у необробленому вигляді.

Як додаткова функція, на основі IP адреси отримувача відбувається виконання функцій спрямованих на визначення мережевої автономної системи (ASN). Адміністратор може використати номер автономної системи при пошуку точок, де спостерігаються втрати, та при розслідуванні мережевих аномалій. Дана функція не блокує головний процес від продовження виконання так як DNS запити можуть виконуватись декілька сотень мілісекунд. Блокування головного процесу на час очікування відповіді від DNS сервера може спричинити затримки через зупинку перехоплення потоків.

У просторі користувача також відбувається етап діагностики і класифікації аномалій. Пошук аномалій відбувається на основі чотирьох правил, які перевіряють всі потоки на основі зібраних діагностичних деталей. Правила перевіряються незалежно, на основі умов відбувається формування результату. Це спрямовано на те, щоб адміністратору вказати на можливі аномалії. Передбачено чотири сценарії, окрім штатної роботи мережі: втрати пакетів під час передачі із підвищеними затримками, втрати пакетів без підвищених затримок, нестабільність шляху або перевантаження буферу, також можлива ситуація, коли точка призначення знаходиться на великій географічній відстані, відповідно значення RTT може коливатись, але без втрат пакетів.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У ході дослідження обґрунтовано, що в умовах стрімкої цифровізації та ускладнення мережевих архітектур, традиційні методи періодичного збору даних стають недостатніми. Перехід до безперервного моніторингу в реальному часі є критично необхідним для забезпечення кібербезпеки, підтримки високої якості обслуговування (QoS) та стабільної роботи інфраструктури в концепції Індустрії 4.0.

Проведений порівняльний аналіз активних та пасивних засобів моніторингу виявив, що пасивний аналіз реального трафіку є більш точним, оскільки не створює додаткового навантаження на канали зв'язку та не спотворює вибірку синтетичними даними. Водночас визначено ключові виклики такого підходу, зокрема необхідність обробки великих обсягів інформації безпосередньо «на льоту» та складність відновлення стану TCP-сесій.

Запропонований метод моніторингу базується на використанні технології eBPF, що дозволяє виконувати фільтрацію та агрегацію метрик безпосередньо у просторі ядра операційної системи. Це вирішує проблему надмірної надлишковості та виснаження ресурсів процесора, притаманну класичним інструментам, які копіюють повні пакети у користувацький простір.

Математична формалізація методу охоплює ідентифікацію потоків за набором ключових параметрів та вимірювання RTT двома способами: через аналіз станів SYN/SYN-ACK та за допомогою опції TCP Timestamp. Такий комбінований підхід забезпечує високу точність вимірювань навіть у складних умовах перевпорядкування пакетів або відсутності підтримки певних опцій мережевим обладнанням.

Програмна реалізація методу, що поєднує мови C та Python, демонструє ефективний розподіл задач: низькорівнева обробка трафіку відбувається у просторі ядра, а складна агрегація, DNS-запити та взаємодія з сервером метрик – у просторі користувача. Це дозволяє мінімізувати копіювання даних між просторами, передаючи лише значущі поля пакетів, що критично важливо для пристроїв з обмеженими ресурсами.

Результати роботи підтверджують, що впровадження інтелектуальної діагностики аномалій на основі зібраних метрик дозволяє не лише констатувати стан мережі, а й оперативно виявляти причини збоїв, такі як перевантаження буферів чи нестабільність маршрутів. Запропонований підхід забезпечує приріст продуктивності та надійність керування сучасною мережевою інфраструктурою.

Міжнародний науково-технічний журнал
«Вимірювальна та обчислювальна техніка в технологічних процесах»
ISSN 2219-9365

References

1. Method for Estimating the Convergence Parameters of Dynamic Routing Protocols in Computer Networks / H. Osukhivska et al. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 22–25 September 2021. 2021. URL: https://doi.org/10.1109/csit52700.2021.9648792.

2. Method of Restoring Parameters of Information Objects in a Unified Information Space Based on Computer Networks / V. Mukhin et al. International Journal of Computer Network and Information Security. 2020. Vol. 12, no. 2. P. 11–21. URL: https://doi.org/10.5815/ijcnis.2020.02.02.

3. A New Static Cost-Effective Parameter for Interconnection Networks of Massively Parallel Computer Systems / M. M. Hafizur Rahman et al. Soft Computing in Data Analytics. Singapore, 2018. P. 147–155. URL: https://doi.org/10.1007/978-981-13-0514-6_15.

4. Pimenta Junior A. P., Abe J. M., Silva G. C. Determination of Operating Parameters and Performance Analysis of Computer Networks with Paraconsistent Annotated Evidential Logic Et. IFIP Advances in Information and Communication Technology. Cham, 2016. P. 3–11. URL: https://doi.org/10.1007/978-3-319-51133-7_1.

5. Kovalenko O. Y., Kuzniuk K. V. Computer network monitoring systems. Mathematical machines and systems. 2023. Vol. 1. P. 50–59. URL: https://doi.org/10.34121/1028-9763-2023-1-50-59.

6. Bodenham D. A., Adams N. M. Continuous Monitoring of a Computer Network Using Multivariate Adaptive Estimation. 2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW), TX, USA, 7–10 December 2013. 2013. URL: https://doi.org/10.1109/icdmw.2013.114.

7. Analysis of Network Parameters Influencing Performance of Hybrid Multimedia Networks / D. Kovac et al. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems. 2013. Vol. 2, no. 3. URL: https://doi.org/10.11601/ijates.v2i3.69.

8. Shete P. J., Awale R. N., Ket S. Y. Channel quality aware cross-layer design based rate adaptive MAC for improving the throughput capacity of multi-hop ad hoc networks. Ad Hoc Networks. 2017. Vol. 63. P. 45–61. URL: https://doi.org/10.1016/j.adhoc.2017.05.009.

9. Kamtam A., Kamar A., Patkar U. C. Artificial Intelligence appreoaches in Cyber Security. International Journal on Recent and Innovation Trends in Computing and Communication. 2016. Vol. 4. Issue 4. Pp. 5-9.

10. Kaur P., Gill N. Performance Comparison of UDP and UDP-Lite for Different Video Codecs. International Journal of Computer Applications. 2012. Vol. 54, no. 12. P. 15–22. URL: https://doi.org/10.5120/8617-2479.

11. eeDTLS: Energy-Efficient Datagram Transport Layer Security for the Internet of Things / U. Banerjee et al. 2017 IEEE Global Communications Conference (GLOBECOM 2017), Singapore, 4–8 December 2017. 2017. URL: https://doi.org/10.1109/glocom.2017.8255053.

Протокол аналізу звіту подібності експертом

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

Автор: Андрій МАРЦЕНЮК

Співавтор:

Назва: Метод та система моніторингу параметрів комп'ютерної мережі в реальному часі на основі одноплатного комп'ютера

Експерт: Катерина БЕРЕЗЬКА

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.26%

Коефіцієнт подібності 2:0.98%

Мікропробіли: 3

Заміна букв: 4

Інтервали: 0

Білі знаки: 6

Дата створення звіту: 2026-04-27 08:07:37.0

Після аналізу Звіту подібності констатую наступне:

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

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

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

Обґрунтування:

2026-04-27 Доцент Андрій Нічепорук

Дата експерт

Mon Apr 27 08:48:00 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

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

Максимальне співпадіння з одним документом 13.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 270694 Назва: МКР Метод та система моніторингу параметрів комп’ютерної мережі в реальному часі на основі одноплатного комп'ютера Додано в БД: 2026-04-27 Автора: Андрій МАРЦЕНЮК Керівники: Катерина БЕРЕЗЬКА Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	133128	1065	19635 (15%)	168 (16%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269834	Назва: Звіт з ПДП Метод та система моніторингу параметрів комп’ютерної мережі в реальному часі на основі одноплатного комп'ютера Додано в БД: 2026-03-17 Автора: Марценюк А. В. Керівники: Капустян М.В Консультанти: Опоненти:	17397 (13.0%)	133 (12.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Андрій МАРЦЕНЮК

Тема: Метод та система моніторингу параметрів комп'ютерної мережі в реальному часі на основі одноплатного комп'ютера

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 80

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано метод та систему моніторингу комп'ютерної мережі в реальному часі на основі одноплатного комп'ютера

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі було розглянуто існуючі методи, розглянуто типові проблеми мережевого моніторингу, окремо була розглянута проблема ефективності використання ресурсів. У другому розділі було визначено типові принцип роботи моніторингу параметрів мережі та архітектуру на які потрібно орієнтуватись щоб забезпечити сумісність та коректність роботи методу із мережевими топологіями. У третьому розділі було запропоновано метод та алгоритм аналізу параметрів та запропоновано ряд вимог, яких необхідно дотримуватись щоб впровадити та використовувати запропонований метод. У четвертому розділі було проведена фіксація та вимірювання кількості системних ресурсів, які необхідні для коректної роботи методу та порівняння метрик із існуючими методами. Також проведене порівняння споживання ресурсів реалізації методу із доступними на ринку інструментами.

4. Позитивні сторони роботи: Запропонований метод і система продемонстрували високу ефективність обробки мережевого трафіку в реальному

часі в пасивному режимі аналізу. Кількість необхідних для методу ресурсів дозволяють виконання на одноплатних комп'ютерах. Позитивною стороною є продумана безпека виконання методу та його реалізації.

5. Негативні сторони роботи: В реалізації методу присутня залежність від ядра операційної системи Linux. Обмежена робота із зашифрованим трафіком.

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на високому рівні.

8. Інші зауваження: –

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «відмінно» 95.00 (А)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Борисов Олександр Володимирович, д.т.н., проф., зав. каф. КН ХНУ

"01" 05 2026р.

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Андрій МАРЦЕНЮК

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-24-1

ЗАЯВА

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

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

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

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

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

Назва кваліфікаційної роботи Метод та система моніторингу параметрів комп'ютерної мережі в реальному часі на основі одноплатного комп'ютера
Автор Андрій МАРЦЕНЮК
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: канд.техн. наук, доцент Катерина БЕРЕЗЬКА

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

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

Підтвердження:

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

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

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

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

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 3,26% і адресується до 75 першоджерела; та системою Anti-Plagiarism складає 13%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

29.04.2026

Завідувач кафедри — Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми — Підпис — Олег САВЕНКО (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи — Підпис — Катерина БЕРЕЗЬКА (Ім'я, ПРІЗВИЩЕ)