ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ ТА МЕРЕЖ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «АЛГОРИТМИ ДЕКОНВОЛЮЦІЇ ДЛЯ ВІДНОВЛЕННЯ ВІДЕОПОТОКІВ ПІСЛЯ ВТРАТ У МЕРЕЖАХ З ОБМЕЖЕНОЮ ПРОПУСКНОЮ ЗДАТНІСТЮ»

на здобуття освітнього ступеня магістра
зі спеціальності 172 Електронні комунікації та радіотехніка
(код, найменування спеціальності)
освітньо-професійної програми Телекомунікаційні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Андрій ЖИТКОВ
(підпис) *(Ім'я ПРІЗВИЩЕ)*

Виконав: здобувач вищої освіти гр. ТСДМ-61
Андрій ЖИТКОВ
Ім'я, ПРІЗВИЩЕ

Керівник: Олександр КРИВОНОСЕНКО
к.т.н., доцент *Ім'я, ПРІЗВИЩЕ*

Рецензент: _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Телекомунікацій

Кафедра ___Телекомунікаційних систем та мереж_____
Ступінь вищої освіти __магістр_____
Спеціальність ___172 Електронні комунікації та радіотехніка _____
Освітньо-професійна програма _____Телекомунікаційні системи та мережі_____

ЗАТВЕРДЖУЮ
Завідувач кафедри __ТСМ_____
_____ Віктор ЗАЇКА
«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Житкову Андрію Олеговичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Алгоритми деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю
керівник кваліфікаційної роботи Олександр КРИВОНОСЕНКО, к.т.н., доцент
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій
від «30» 10. 2025 р. № 467

2. Строк подання кваліфікаційної роботи «___» _____ 2025 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Підвищення пропускної здатності та зменшення затримки.
2. Усунення втрат в мережі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Обмеження репрезентативної здатності та оцінка помилки узагальнення для мереж.
2. Оптимізація веб-серверів для підвищення пропускної здатності та зменшення затримки.
3. Використання інфраструктурного компоненту Kafaka Connect.
4. Підвищення пропускної здатності для усунення втрат в мережі.

5. Перелік ілюстративного матеріалу: ___презентація на ____слайдах_____

6. Дата видачі завдання «___» _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	20.10.2025	Викон.
2	Розробка першого розділу	27.10.2025	Викон.
3	Розробка другого розділу	07.11.2025	Викон.
4	Розробка третього розділу	17.11.2025	Викон.
5	Розробка четвертого розділу	27.11.2025	Викон.
6	Висновки, вступ, реферат	07.12.2025	Викон.
7	Розробка презентації	17.12.2025	Викон.

Здобувач вищої освіти _____ Андрій ЖИТКОВ
(підпис) *(Ім'я ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Олександр КРИВОНОСЕНКО
(підпис) *(Ім'я ПРІЗВИЩЕ)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут телекомунікацій
ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Житков А.О. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 172 Електронні комунікації та радіотехніка
(код, найменування спеціальності)
освітньо-професійної програми Телекомунікаційні системи та мережі
(назва)
на тему: «Алгоритми деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю»

Кваліфікаційна робота і рецензія додаються.

Директор ННІТ _____ Владислав КРАВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Житков Андрій Олегович показав гарну теоретичну та інженерну підготовку, уміння володіти новими комп'ютерними технологіями, користуватися навчальною, довідковою і науково-технічною літературою. Працюючи над завданнями, які були поставлені керівником, проявив ініціативність, сумлінність та хист до наукової роботи.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Житкова Андрія Олеговича на оцінку «відмінно» та присвоїти йому кваліфікацію магістр з електронних комунікацій та радіотехніки.

Керівник кваліфікаційної роботи _____ Олександр КРИВОНОСЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 20___ року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Житков А.О. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри телекомунікаційних систем та мереж _____ Віктор ЗАЇКА
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 20___ року

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня магістра

здобувача вищої освіти Житкова Андрія Олеговича
(прізвище, ім'я, по батькові)
на тему «Алгоритми деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю»

Актуальність.
Зростання обсягів відеотрафіку в сучасних телекомунікаційних мережах зумовлює підвищені вимоги до якості та надійності передавання даних. В умовах обмеженої пропускної здатності та нестабільних мережевих характеристик втрати пакетів і затримки призводять до суттєвої деградації якості відеопотоків. Традиційні мережеві механізми не завжди забезпечують необхідний рівень якості сервісу, що актуалізує використання алгоритмів деконволюції та інтелектуальної обробки відео на прикладному рівні. Поєднання таких методів з оптимізацією серверної та мережевої інфраструктури дозволяє підвищити стійкість відеопотоків і якість їх відтворення в реальних умовах експлуатації мереж.
Позитивні сторони.
1. У роботі використано комплексний підхід, що поєднує мережеві, серверні та прикладні методи підвищення якості відеопотоків.
2. В роботі виконано ґрунтовний аналіз проблем передавання відеопотоків у мережах з обмеженою пропускною здатністю та нестабільними параметрами каналу.
Недоліки.
1. Зв'язок між теоретичними моделями графових нейронних мереж і практичною задачею відновлення відеопотоків подано недостатньо формалізовано.
2. Експериментальна частина могла б бути розширена шляхом використання більшої кількості сценаріїв навантаження та показників якості відео.

Відзначені зауваження не впливають на загальну позитивну оцінку магістерської кваліфікаційної роботи

Висновок: *кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "відмінно", а здобувач Житков Андрій Олегович заслуговує присвоєння кваліфікації: магістр з електронних комунікацій та радіотехніки*.

Рецензент
науковий ступінь, вчене звання _____ (підпис) _____ *(Ім'я ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 81 стор., 24 рис., 3 табл., 25 джерел.

Мета роботи – підвищення якості відновлення відеопотоків у телекомунікаційних мережах з обмеженою пропускною здатністю шляхом застосування алгоритмів деконволюції та оптимізації серверної й мережевої інфраструктури.

Об’єкт дослідження – процес передавання та обробки відеопотоків у телекомунікаційних мережах з обмеженими та нестабільними параметрами пропускної здатності.

Предмет дослідження – методи та алгоритми деконволюційного відновлення відеопотоків, а також способи оптимізації серверних і мережевих ресурсів, що впливають на якість відтворення відео після втрат даних.

Короткий зміст роботи: у кваліфікаційній роботі досліджено алгоритми деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю з урахуванням як алгоритмічних, так і мережевих аспектів передавання даних. Показано, що деградація якості відео в сучасних мережах не може бути ефективно усунена виключно на одному рівні та потребує комплексного підходу. Встановлено, що навіть у мережах доступу на основі стандарту DOCSIS 3.1, які використовують OFDM/OFDMA та LDPC, залишаються втрати пакетів і коливання затримок. Обґрунтовано роль алгоритмів деконволюції у відновленні просторово-часової структури відео та зменшенні візуальних артефактів. Окрему увагу приділено оптимізації потокової інфраструктури передавання даних на основі Kafka, що забезпечує стабільне надходження відеоданих і підвищує ефективність їх подальшої обробки.

КЛЮЧОВІ СЛОВА: АЛГОРИТМИ ДЕКОНВОЛЮЦІЇ, ВІДНОВЛЕННЯ ВІДЕОПОТОКІВ, ОБМЕЖЕНА ПРОПУСКНА ЗДАТНІСТЬ, ВТРАТИ ПАКЕТІВ, ЯКІСТЬ ВІДЕО, DOCSIS 3.1, OFDM/OFDMA, LDPC, KAFKA, ПОТОКОВА ОБРОБКА ДАНИХ, ОПТИМІЗАЦІЯ МЕРЕЖІ, СЕРВЕРНА ІНФРАСТРУКТУРА

ABSTRACT

Text part of the master's qualification work: 81 pages, 24 figures, 3 table, 25 sources.

The purpose of the work is to improve the quality of video stream restoration in telecommunication networks with limited bandwidth by applying deconvolution algorithms and optimizing server and network infrastructure.

Object of the study is the process of transmitting and processing video streams in telecommunication networks with limited and unstable bandwidth parameters.

Subject of the study is methods and algorithms for deconvolution restoration of video streams, as well as methods for optimizing server and network resources that affect the quality of video playback after data loss.

Summary of the work: the qualification work investigates deconvolution algorithms for restoring video streams after losses in networks with limited bandwidth, taking into account both algorithmic and network aspects of data transmission. It is shown that video quality degradation in modern networks cannot be effectively eliminated exclusively at one level and requires a comprehensive approach. It is found that even in access networks based on the DOCSIS 3.1 standard, which use OFDM/OFDMA and LDPC, packet loss and delay fluctuations remain. The role of deconvolution algorithms in restoring the spatio-temporal structure of video and reducing visual artifacts is substantiated. Special attention is paid to the optimization of the Kafka-based streaming data transmission infrastructure, which ensures stable video data arrival and increases the efficiency of their further processing.

KEYWORDS: DECONVOLUTION ALGORITHMS, VIDEO STREAM RECOVERY, LIMITED BANDWIDTH, PACKET LOSS, VIDEO QUALITY, DOCSIS 3.1, OFDM/OFDMA, LDPC, KAFKA, STREAMING DATA PROCESSING, NETWORK OPTIMIZATION, SERVER INFRASTRUCTURE

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1 ОБМЕЖЕННЯ РЕПРЕЗЕНТАТИВНОЇ ЗДАТНОСТІ ТА ОЦІНКА ПОМИЛКИ УЗАГАЛЬНЕННЯ ДЛЯ МЕРЕЖ 11
1.1 Графові нейронні мережі в телекомунікаціях 11
1.2 Обмеження в графових системах 15
1.3 Крок до більш потужних GNN. Оцінка помилки узагальнення 19
РОЗДІЛ 2 ОПТИМІЗАЦІЯ ВЕБ-СЕРВЕРІВ ДЛЯ ПІДВИЩЕННЯ ПРОПУСКНОЇ ЗДАТНОСТІ ТА ЗМЕНШЕННЯ ЗАТРИМКИ 25
2.1 Обладнання та основні характеристики 26
2.2 Огляд відновлювальних алгоритмів 41
РОЗДІЛ 3 ВИКОРИСТАННЯ ІНФРАСТРУКТУРНОГО КОМПОНЕНТУ KAFAKA CONNECT 52
3.1 Основні налаштування 52
3.2 JMX метрики 57
РОЗДІЛ 4 ПІДВИЩЕННЯ ПРОПУСКНОЇ ЗДАТНОСТІ ДЛЯ УСУНЕННЯ ВТРАТ В МЕРЕЖІ 67
4.1 Специфікація 67
4.2 Максимальна пропускна здатність для запобігання втрат 73
4.3 Upstream 78
ВИСНОВКИ 80
ПЕРЕЛІК ПОСИЛАНЬ 82
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 84

ВСТУП

Відеопотоки є одним із найвибагливіших до якості передавання типів даних у сучасних телекомунікаційних мережах. Зростання обсягів мультимедійного трафіку, поширення потокового відео високої роздільної здатності та реального часу, а також активне використання бездротових і кабельних мереж доступу призводять до того, що проблема втрат даних і деградації якості відео набуває особливої актуальності. У мережах з обмеженою або динамічною пропускною здатністю втрати пакетів, коливання затримок і нерівномірність передавання неминуче впливають на цілісність відеопотоків і сприйняття якості сервісу кінцевими користувачами.

Традиційні мережеві механізми підвищення надійності, зокрема корекція помилок і адаптивні методи модуляції, хоча й дозволяють зменшити кількість втрат, не здатні повністю усунути наслідки деградації каналу. Навіть у сучасних мережах доступу, таких як кабельні мережі на основі стандарту DOCSIS 3.1, що використовують OFDM/OFDMA та алгоритми LDPC, залишкові втрати та спотворення відеоданих залишаються актуальною проблемою, особливо за пікових навантажень або в гетерогенних мережевих середовищах.

У зв’язку з цим дедалі більшого значення набувають методи обробки відео на прикладному рівні, зокрема алгоритми деконволюції, які дозволяють відновлювати втрачені або спотворені фрагменти відеопотоків, компенсувати розмиття та зменшувати візуальні артефакти. Такі алгоритми використовують просторово-часові залежності між кадрами і здатні істотно підвищувати суб’єктивну якість відео навіть за несприятливих умов передавання. Ефективність деконволюційних методів, однак, значною мірою залежить від стабільності надходження відеоданих і правильного функціонування всієї інфраструктури передавання.

Важливу роль у цьому контексті відіграють потокові архітектури обробки даних і системи надійної доставки, зокрема платформи на основі Apache Kafka та Kafka Connect. Вони забезпечують масштабовану та відмовостійку передачу фрагментів відеопотоків і метаданих, створюючи основу для подальшої

алгоритмічної обробки та відновлення відео. Правильне налаштування таких систем дозволяє зменшити затримки, уникнути вузьких місць і забезпечити умови, за яких алгоритми деконволюції можуть працювати максимально ефективно.

Таким чином, тема даної кваліфікаційної роботи, присвяченої алгоритмам деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю, є актуальною з наукової та практичної точок зору. У роботі розглядається комплексний підхід до проблеми відновлення відео, який поєднує особливості сучасних мереж доступу, потокову інфраструктуру передавання даних і алгоритмічні методи обробки відеопотоків з метою підвищення якості сервісу в реальних умовах експлуатації мереж.

1 ОБМЕЖЕННЯ РЕПРЕЗЕНТАТИВНОЇ ЗДАТНОСТІ ТА ОЦІНКА ПОМИЛКИ УЗАГАЛЬНЕННЯ ДЛЯ МЕРЕЖ

1.1 Графові нейронні мережі в телекомунікаціях

Графові нейронні мережі (Graph Neural Networks, GNN) — це клас моделей машинного навчання, призначених для безпосередньої роботи з графовими структурами. Вони дають змогу враховувати не лише ознаки окремих вершин, а й топологію графа, тобто зв’язки та взаємодію між елементами.

Типова GNN складається з невеликої кількості послідовно з’єднаних шарів, на кожному з яких відбувається ітеративне оновлення представлень вершин шляхом агрегації інформації від їхніх сусідів. Такий процес часто інтерпретується як узагальнена згортка на графі, а зворотна операція — деконволюція — може застосовуватися для відновлення або поширення узагальнених ознак з високорівневих представлень назад до елементів графа, наприклад у задачах реконструкції структури чи генерації графів.

До найпоширеніших архітектур графових нейронних мереж належать GCN (Graph Convolutional Network), GraphSAGE, GAT (Graph Attention Network) та GIN (Graph Isomorphism Network), кожна з яких по-різному реалізує механізми агрегації, згортки та, за потреби, деконволюційного відображення інформації в графових просторах.

Процес оновлення ембеддингів вершин для будь-якої GNN-архітектури можна узагальнити двома формулами:

$$a_v^{t+1} = \mathrm{AGG}\left(h_w^t : w \in \mathcal{N}(v)\right) \tag{1.1}$$

$$h_v^{t+1} = \mathrm{COMBINE}(h_v^t, a_v^{t+1}) \tag{1.2}$$

де AGG - зазвичай функція, інваріантна до перестановок (sum, mean, max etc.), COMBINE - функція, що поєднує уявлення вершини та її сусідів.

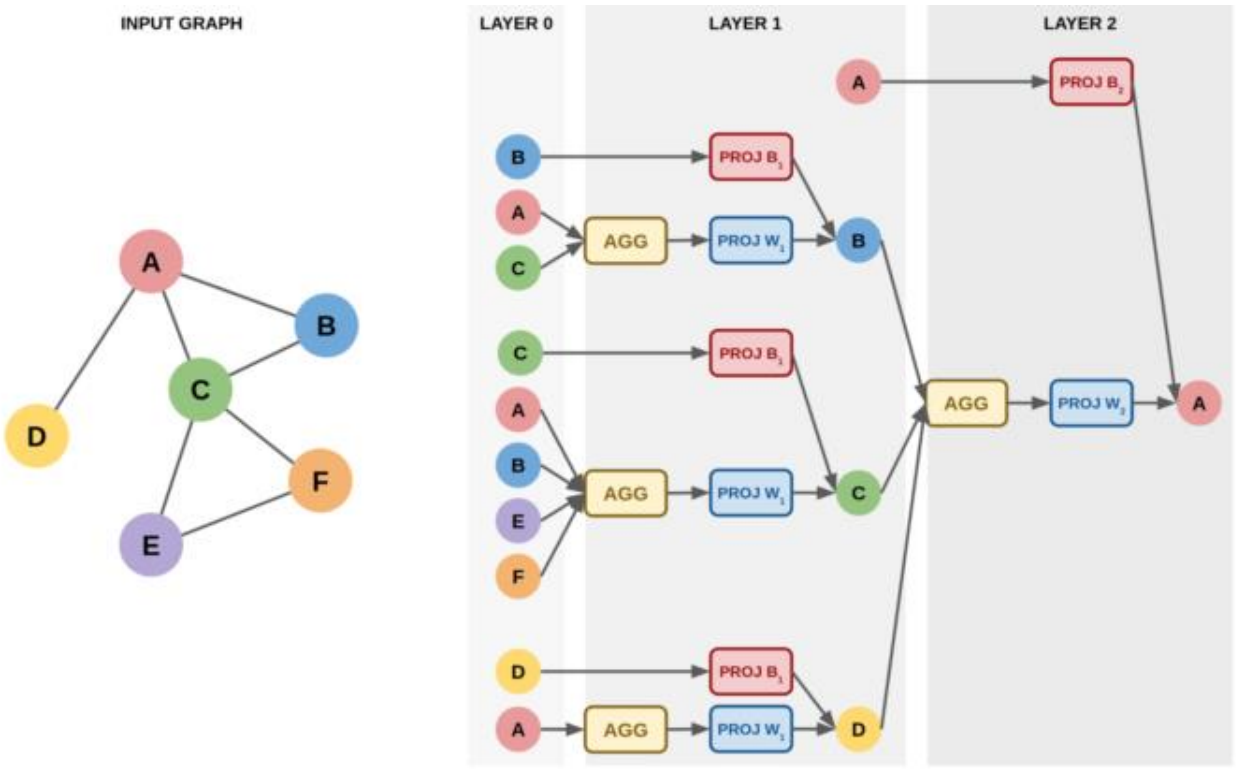


Рис. 1.1 Дерево обчислень для двошарової GNN з прикладу вузла A

Більш просунуті архітектури графових нейронних мереж здатні враховувати додаткову структурну та атрибутивну інформацію графа. Окрім ознак вершин, до моделі можуть бути включені характеристики ребер, геометричні параметри, такі як кути між ребрами, ваги та напрямленість зв'язків, а також інші контекстні властивості. Це дає змогу точніше моделювати взаємодію між елементами графа та підвищувати якість аналізу складних структурованих даних.

У розділі розглядається клас графових нейронних мереж, орієнтованих на розв'язання задачі класифікації графів (graph classification). Загальна архітектура таких моделей складається з кількох послідовних етапів. На першому етапі формуються векторні представлення вершин (ембеддинги) шляхом застосування L шарів графових згорток. У процесі кожної ітерації відбувається агрегація інформації від сусідніх вершин, що дозволяє накопичувати як локальні, так і глобальні структурні ознаки графа.

На другому етапі ембеддинги окремих вершин узагальнюються в єдине представлення всього графа за допомогою функцій агрегації, інваріантних до перестановок вершин, таких як сума (sum), середнє значення (mean) або максимум

(max). Такий підхід забезпечує незалежність результату від порядку нумерації вершин і коректне відображення структури графа в компактному векторному вигляді.

Крім задач класифікації, подібні архітектурні принципи можуть застосовуватися і в суміжних прикладних задачах телекомунікацій. Зокрема, механізми деконволюції (зворотної згортки) можуть використовуватися для відновлення втрачених або спотворених даних у відеопотоках, що передаються через мережі з обмеженою пропускною здатністю. У таких умовах втрати пакетів і обмеження швидкості передачі призводять до деградації якості відео, а деконволюційні шари дозволяють відновлювати просторово-часові характеристики сигналу на основі узагальнених ознак, отриманих під час згорткової обробки.

На завершальному етапі побудоване представлення графа або відновлений сигнал подається на вхід відповідного класифікатора чи реконструкційного модуля, який формує кінцевий результат — бінарну мітку класу або відновлений відеопотік. Таким чином, поєднання графових згорток, інваріантних механізмів агрегації та деконволюції створює основу для ефективного аналізу й реконструкції даних у сучасних мережах передачі інформації з обмеженими ресурсами.

Обмеження графових нейронних мереж

У цьому розділі розглядаються основні обмеження графових нейронних мереж, характерні для кількох ключових класів GNN. Проаналізовано, які структурні та теоретичні обмеження виникають під час обробки графових даних і як вони впливають на здатність моделей розрізняти складні графові структури.

- **Локально інваріантні графові нейронні мережі (LU-GNN).** До цього класу належать такі поширені архітектури, як GCN, GraphSAGE, GAT, GIN та інші подібні моделі. Їхньою характерною особливістю є інваріантність до перестановок сусідніх вершин, що досягається шляхом симетричних функцій агрегації (сума, середнє, максимум). Водночас така інваріантність накладає суттєві обмеження: моделі не здатні враховувати порядок або взаємне розташування сусідів, що призводить до втрати частини структурної інформації. У результаті різні графи можуть мати однакові

представлення, навіть якщо вони відрізняються за топологією або локальними шаблонами зв’язків.

- **CPNGNN з локальним упорядкуванням сусідів.**

  У цьому класі мереж вводиться локальний порядок вершин шляхом нумерації сусідів кожної вершини від 1 до d, де d — ступінь відповідної вершини. Такий підхід, відомий в оригінальних роботах як *port numbering*, дозволяє частково подолати обмеження локальної інваріантності та враховувати позиційні відмінності між сусідами. Однак цей метод має власні недоліки, зокрема залежність від способу нумерації, яка може бути неунікальною або складною для узгодження в реальних графах. Крім того, введення порядку збільшує складність моделі та ускладнює її узагальнення на нові графові структури.

- **DimeNet для тривимірних графів.**

  Архітектура DimeNet орієнтована на роботу з 3D-графами та враховує не лише топологічні зв’язки, а й метричну інформацію, зокрема відстані між вершинами та кути між ребрами. Це дозволяє точніше моделювати фізичні та геометричні властивості систем, наприклад у задачах молекулярного моделювання. Водночас така деталізація призводить до зростання обчислювальної складності та вимог до обсягу даних. Крім того, моделі цього класу менш універсальні, оскільки їх застосування обмежене графами з чітко визначеною просторовою інтерпретацією.

Таким чином, у цьому розділі показано, що кожен клас графових нейронних мереж має власні обмеження, пов’язані з інваріантністю, способом кодування локальної структури та складністю урахування геометричних характеристик. Розуміння цих обмежень є важливим для коректного вибору архітектури GNN залежно від конкретної прикладної задачі.

1.2 Обмеження в графових системах

Обмеження LU-GNN

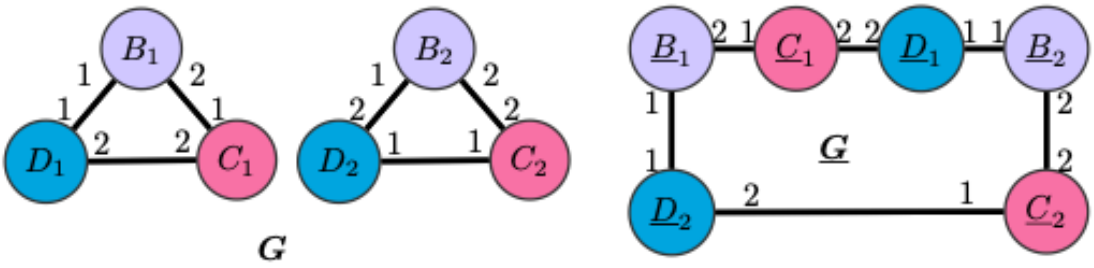


Рис. 1.2 Приклад графа G з різними варіантами локального упорядкування вершин і ребер (port numbering), що демонструє неоднозначність представлення структури при однаковій топології

Графи (G) та (G') є нерозрізнюваними для локально інваріантних графових нейронних мереж (LU-GNN), оскільки відповідні обчислювальні дерева для вершин одного й того самого кольору в обох графах повністю збігаються. У результаті цього процес агрегації інформації від сусідніх вершин відбувається однаковим чином, а векторні представлення (ембеддинги), сформовані за допомогою readout-функцій, інваріантних до перестановок вершин, також виявляються ідентичними. Це означає, що така модель не здатна відрізнити ні окремі вершини однакового кольору, ні самі графи між собою, попри наявні структурні відмінності.

Водночас графові нейронні мережі класу CPNGNN, у яких вводиться відповідний локальний порядок сусідніх вершин, мають значно більшу розрізнювальну здатність. Завдяки використанню нумерації сусідів (port numbering) модель отримує додаткову інформацію про позиційне розташування вершин у локальному оточенні. Це дозволяє їй коректно розрізняти як однотипні за кольором вершини в графах (G) та (G'), так і самі графи в цілому. Таким чином, запровадження локального порядку усуває обмеження, притаманні LU-GNN, і

підвищує виразну потужність графових нейронних мереж при аналізі складних структур.

Обмеження CPNGNN

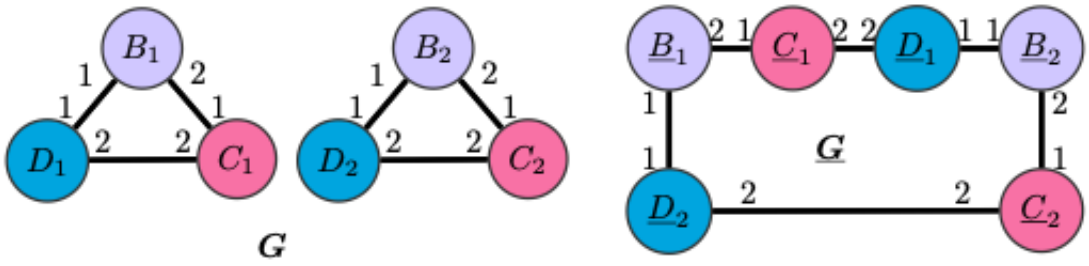


Рис. 1.3 Ілюстрація двох ізоморфних графів GGG з однаковою топологією, але різним локальним упорядкуванням сусідніх вершин (port numbering), що впливає на здатність графових нейронних мереж розрізняти їх структуру

Проте, існує “поганий” порядок, у якому CPNGNN зробить однакові ембеддинги для вершин одного кольору.

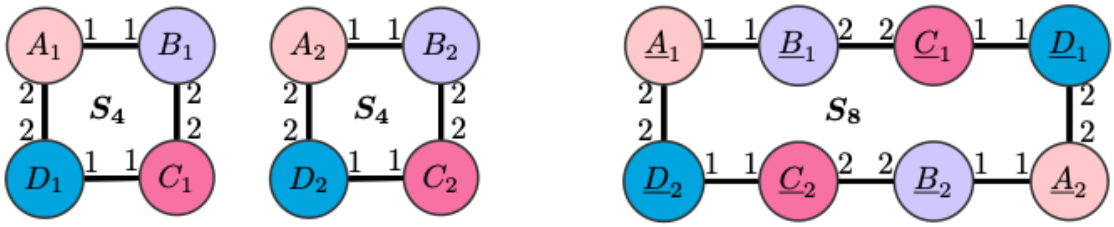


Рис. 1.4 Приклад графів S_4 та S_8, що демонструють вплив локальної симетрії та нумерації портів на розрізнювальну здатність графових нейронних мереж при аналізі вершин і структури графа

Граф S_8 та дві копії графа S_4 відрізняються між собою за низкою важливих структурних характеристик. Зокрема, вони мають різний обхват і окружність, тобто різну довжину найкоротшого та найдовшого циклів відповідно, а також відрізняються за радіусом, діаметром і загальною кількістю циклів у графі. Ці параметри безпосередньо відображають глобальну топологію графа та відіграють суттєву роль при його аналізі й класифікації.

Однак навіть за наявності таких відмінностей, графова нейронна мережа класу CPNGNN з фіксованим локальним порядком портів і використанням readout-

функції, інваріантної до перестановок вершин, сформує для цих графів однакові векторні представлення (ембеддинги). Це означає, що на етапі узагальнення інформації на рівні всього графа модель втрачає здатність розрізняти їх глобальні структурні особливості.

У результаті на основі отриманих ембеддингів стає неможливо коректно відновити або оцінити справжні характеристики кожного з графів окремо. Таким чином, навіть попри введення локального порядку в CPNGNN, використання інваріантного до перестановок механізму агрегації накладає обмеження на здатність моделі відображати глобальні топологічні властивості графів, що слід враховувати при розробці та застосуванні графових нейронних мереж для задач аналізу складних структур.

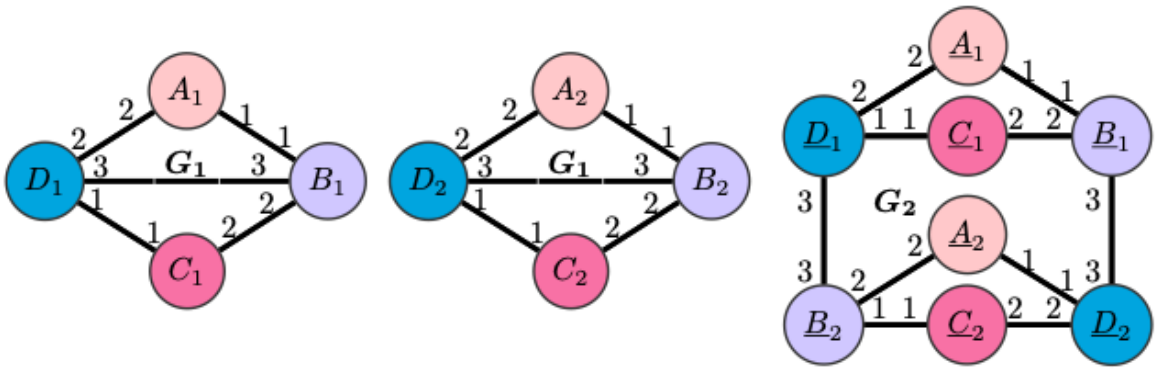


Рис. 1.5 Приклад графів G_1 та G_2 з різною глобальною структурою та кількістю циклів, які, попри топологічні відмінності, можуть мати однакові ембеддинги при використанні CPNGNN з фіксованим порядком портів і readout-функцією, інваріантною до перестановок

За наведених умов графова нейронна мережа класу CPNGNN не здатна відновити структурні характеристики, розглянуті в попередньому прикладі, ані для графа G_2, ані для двох копій графа G_1. Причина полягає в тому, що навіть за наявності фіксованого локального порядку портів така модель, у поєднанні з readout-функцією, інваріантною до перестановок, формує однакові або надто подібні ембеддинги для графів із різною глобальною геометрією. Унаслідок цього важливі топологічні та просторові відмінності між графами втрачаються на етапі узагальнення.

На відміну від цього, архітектура DimeNet здатна коректно розрізняти такі графи та відновлювати їхні характеристики. Це пояснюється тим, що DimeNet безпосередньо враховує геометричну інформацію, зокрема кути між ребрами та відстані між вершинами. У розглянутому випадку кути в різних частинах графів не всюди збігаються, що створює додаткові розрізнювальні ознаки для моделі. Наприклад, кут $\angle A_1B_1C_1$ у графі G_1 не є тотожним куту $\angle A_1'B_1'C_1'$ у графі G_2, попри подібність їх локальних зв'язків.

Таким чином, урахування кутових залежностей і просторової конфігурації вершин дозволяє DimeNet зберігати та використовувати інформацію, недоступну для CPNGNN. Це забезпечує вищу виразну здатність моделі та робить її більш придатною для задач, у яких важливими є не лише топологічні, а й геометричні властивості графів.

Обмеження DimeNet

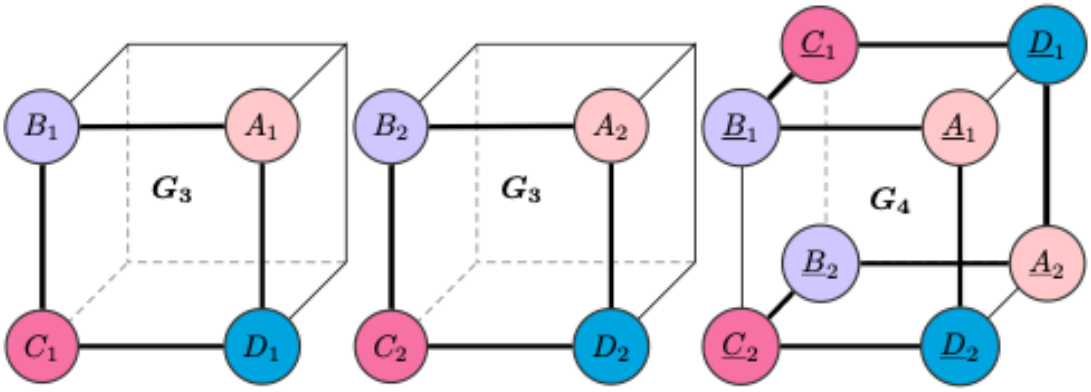


Рис. 1.6 Приклад графів G_3 та G_4 з різною просторовою конфігурацією вершин, що ілюструє вплив геометричної структури на розрізнювальну здатність графових нейронних мереж, зокрема при врахуванні просторових залежностей і кутів між ребрами

Модель DimeNet не здатна розрізнити граф G_4 і граф, утворений з двох копій G_3, оскільки в обох випадках відстані між відповідними вершинами та кути між ребрами є повністю ідентичними. За таких умов геометричні ознаки, на які спирається DimeNet, не містять розрізнювальної інформації, унаслідок чого модель формує однакові або еквівалентні векторні представлення для обох графів.

У результаті DimeNet не може коректно обчислити або відновити структурні характеристики кожного з графів окремо, незважаючи на їх відмінності з точки зору глобальної топології. Це свідчить про те, що навіть урахування геометрії не гарантує повної виразної здатності моделі в усіх можливих випадках.

Крім того, можна зауважити, що графи G_4 та G_3 відповідають графам S_4 і S_8, натягнутим на кубічну структуру. Така просторово регулярна конфігурація призводить до вирівнювання як відстаней, так і кутів між ребрами. Отже, навіть удосконалена версія DimeNet з додатковою нумерацією портів, успадкованою з графів S_4 та S_8, не зможе ефективно впоратися з цією задачею. Це підкреслює фундаментальні обмеження сучасних графових нейронних мереж, які зберігаються навіть при поєднанні топологічної та геометричної інформації.

1.3 Крок до більш потужних GNN. Оцінка помилки узагальнення

Запропоновано об’єднати ідеї розглянутих вище підходів і доповнити їх розширеним набором геометричних характеристик. Такий підхід передбачає більш повне використання просторової інформації з метою підвищення розрізнювальної здатності графових нейронних мереж та подолання обмежень, притаманних існуючим архітектурам.

Зокрема, окрім традиційного врахування відстаней між вершинами та кутів між окремими ребрами, доцільно включати додаткові геометричні ознаки, пов’язані з взаємним розташуванням площин у графі. Такі площини можуть задаватися трійками ребер, що дозволяє описувати більш складні просторові взаємозв’язки між елементами графа. Урахування кутів між площинами та відповідних відстаней дає змогу захопити вищі порядки геометричних залежностей, які не відображаються при аналізі лише парних або трійкових взаємодій.

Таким чином, розширення просторового опису графів за рахунок багатовимірних геометричних характеристик створює передумови для побудови більш виразних і універсальних моделей, здатних точніше відображати як локальну, так і глобальну структуру складних графових об’єктів.

Запропонована GNN працює так:

Застосування шару DimeNet.

На цьому етапі виконується базова обробка графа з урахуванням геометричних залежностей, зокрема відстаней і кутів між елементами структури.

Розширення message-ембеддинга ребра геометричною інформацією.

Message-ембеддинг ребра $m^t_(w,v)$ доповнюється геометричними ознаками Ф_(w,v), що дозволяє врахувати просторові взаємозв'язки між вершинами. Оновлення здійснюється за формулою:

$$m_{wv}^{(l)} = f!\left(m_{wv}^{(l)}, \Phi_{wv}\right) \tag{1.3}$$

Формування ембеддинга вершини з урахуванням локального порядку.

Представлення вершини обчислюється з урахуванням введеного локального порядку сусідів ($c_v(i)$, $t_{i,v}$) — i-й сусід вершини v, а $t_{i,v}$ — ембеддинг порядку. Формула оновлення представлення вершини має вигляд:

$$h_v^{(l+1)} = f\left(h_v^{(l)}, m_{c_v(1),v}^{(l)}, t_{1,v}, \dots, m_{c_v(d(v)),v}^{(l)}, t_{d(v),v}\right) \tag{1.4}$$

Readout-етап, інваріантний до перестановок.

Отримані ембеддинги вершин агрегуються в єдине представлення графа за допомогою симетричних функцій агрегації, незалежних від порядку вершин.

Вихідний модуль.

На фінальному етапі використовується класифікатор або регресор, залежно від типу задачі, для формування кінцевого результату.

Оцінка помилки узагальнення

Розглядається конкретний клас моделей: LU-GNN, в якій оновлення уявлення вершини відбувається згідно:

$$h_v^{t+1} = \phi!\left(W_1 x_v + W_2 \rho!\left(\sum_{u \in \mathcal{N}(v)} g(h_u^t)\right)\right) \tag{1.5}$$

Тут φ, g та ρ є нелінійними функціями активації, а x_v — вектор ознак, що відповідає вершині v. Вважається, що функція ρ задовольняє умову ρ(0) = 0. Для всіх вершин v виконується обмеження на норму вхідних ознак $\|x_v\|_2 \le B_x$, а для всіх

$x \in \mathbb{R}^d$ значення функції активації φ є обмеженими, тобто $\|\varphi(x)\|_\infty \leq \delta < \infty$. Крім того, виконуються умови $\varphi(0) = 0$ та $g(0) = 0$, що гарантує коректну поведінку моделі за нульових вхідних значень.

Також припускається, що функції φ, g і ρ є ліпшицевими з відповідними константами $C\varphi$, Cg та $C\rho$. Матриці ваг W_1 і W_2 обмежені за нормою, а саме $\|W_1\|_2 \leq B_1$ та $\|W_2\|_2 \leq B_2$, що забезпечує стабільність навчання та обмеження зростання значень ембеддингів.

Далі до представлення кожної вершини застосовується бінарний класифікатор окремо. Отримані для всіх вершин передбачення усереднюються, формуючи загальний результат для всього графа. Параметр β бінарного класифікатора також має обмеження за нормою, зокрема виконується умова $\|\beta\|_2 \leq B\beta$.

Нехай $f(G)$ — результат застосування графової нейронної мережі до графа G, а $y \in \{0, 1\}$ — істинна мітка цього графа. Тоді величина $p(f(G), y) = y(2f(G) - 1) + (1 - y)(1 - 2f(G))$ інтерпретується як різниця між імовірностями правильного та неправильного передбачення мітки. Значення $p(f(G), y) < 0$ виникає лише у випадку помилки класифікації, що дозволяє використовувати цю величину як зручну міру якості роботи моделі.

Тоді емпірична складність Радемахера для класу функцій прогнозування графової нейронної мережі f на тренувальній вибірці визначається для набору графів і відповідних міток $\{G_j, y_j\}_{j=1}^n$ і характеризує здатність моделі узгоджуватися з випадковими мітками на скінченній вибірці.

$$\hat{\mathcal{R}}(f) = \frac{1}{m}\sum_{j=1}^{m} \text{loss}_\gamma\,!\left(-p!\left(f\left(G_j\right), y_j\right)\right) \tag{1.6}$$

У роботі запропоновано побудову доказу на основі спеціальної леми, в якій стверджується, що з високою ймовірністю похибку передбачення графової нейронної мережі можна обмежити сумою двох складових. Перша складова відповідає помилці на тренувальній вибірці, тоді як друга характеризує здатність

моделі коректно визначати клас графа за його ембеддингом. Такий підхід дозволяє розділити вплив якості навчальних даних і власних виразних можливостей моделі на загальну помилку прогнозування.

Для класу моделей, який розглядається в роботі, а саме графових нейронних мереж, що визначають мітку всього графа на основі міток більшості його вершин, друга складова може бути додатково спрощена. Замість аналізу здатності передбачення за глобальним ембеддингом графа її можна замінити на здатність моделі прогнозувати клас за окремими вершинами. У такому випадку аналіз зводиться до вивчення обчислювальних дерев, що відповідають локальним околицям вершин, оскільки саме ці дерева визначають формування вершинних представлень у GNN.

Усі наведені міркування виконуються з метою обґрунтування можливості використання двох ключових фактів. По-перше, оскільки розглядається дерево, ембеддинг вершини можна подати у рекурсивному вигляді через ембеддинги її піддерев. Це суттєво спрощує теоретичний аналіз, адже дозволяє послідовно описувати вплив локальних структур на кінцеве представлення вершини. По-друге, показано, що малі зміни у вагових коефіцієнтах моделі призводять лише до малих змін у результаті класифікації. Відповідне доведення наведене в роботі та є важливим для обґрунтування стабільності й узагальнювальної здатності графових нейронних мереж.

Описані теоретичні результати мають значення не лише для задач класифікації графів, а й для прикладних задач у телекомунікаційних системах. Зокрема, подібні ідеї можуть застосовуватися при використанні згорткових і деконволюційних механізмів для відновлення відеопотоків після втрат пакетів у мережах з обмеженою пропускною здатністю. У таких умовах локальні спотворення або пропуски даних можна інтерпретувати як порушення в обчислювальних «деревах» просторово-часових залежностей. Деконволюційні шари, у поєднанні зі стабільністю моделей до малих змін параметрів, дозволяють відновлювати втрачені фрагменти відео на основі контексту сусідніх кадрів і просторових структур, аналогічно до того, як GNN відновлює інформацію про

вершину з її локального оточення. Це підкреслює універсальність розглянутих підходів і можливість їх застосування в задачах реконструкції мультимедійних потоків у мережах з обмеженими ресурсами.

Це дозволяє вивести межі помилки узагальнення за аналогією з RNN.

$< 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd}{\sqrt{m}\gamma}\right)$	$\tilde{\mathcal{O}}\left(\frac{r}{\sqrt{m}\gamma}\right)$
$= 1/d$	$\tilde{\mathcal{O}}\left(\frac{rdL}{\sqrt{m}\gamma}\right)$	$\tilde{\mathcal{O}}\left(\frac{rL}{\sqrt{m}\gamma}\right)$
$> 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd\sqrt{rL}}{\sqrt{m}\gamma}\right)$	$\tilde{\mathcal{O}}\left(\frac{r\sqrt{rL}}{\sqrt{m}\gamma}\right)$

Рис. 1.7 Порівняння з RNN

У таблиці С наведено оцінку складності перенавчання, яка задається у вигляді $C = C\varphi \cdot Cg \cdot C\rho \cdot B_2$, де r — розмір ембеддинга, d — максимальний ступінь вершини графа, m — обсяг тренувальної вибірки, L — кількість шарів моделі, а γ — зазор, що залежить від конкретних даних. Така параметризація дозволяє явно пов’язати складність моделі з її архітектурними характеристиками та властивостями навчальних даних, що є важливим для практичного аналізу поведінки графових нейронних мереж.

Отримані оцінки дають змогу значно точніше оцінювати помилку узагальнення порівняно з попередніми підходами. Зокрема, вони є суттєво більш інформативними, ніж результати, в яких помилка оцінювалася порядком $O(r^3N / \sqrt{m})$. Удосконалений аналіз дозволяє краще враховувати вплив розміру ембеддинга, кількості шарів і структурних властивостей графа, що безпосередньо відображається на якості теоретичних гарантій.

Ключовим елементом цієї роботи є використання локальної інваріантності до перестановок усередині графових нейронних мереж, а не лише на етапі глобальної агрегації (readout), як це часто робиться в інших підходах. Саме цей аспект принципово відрізняє розглянуту модель від попередніх робіт. Завдяки такому підходу вдається уникнути необхідності врахування всіх можливих порядків, у яких

можуть бути розташовані вершини графа, що суттєво спрощує аналіз і водночас підвищує точність оцінки помилки узагальнення.

Особливу увагу приділено балансу між виразною потужністю моделі, тобто її здатністю розрізняти максимально широкий клас графових структур, і здатністю до узагальнення. Хоча більш складні архітектури зазвичай мають вищу розрізнювальну здатність, вони також є більш схильними до перенавчання. Тому на практиці доцільно обирати такі моделі, які забезпечують компроміс між складністю та стабільністю навчання, з урахуванням розміру датасета, доступних обчислювальних ресурсів і специфіки прикладної задачі.

2 ОПТИМІЗАЦІЯ ВЕБ-СЕРВЕРІВ ДЛЯ ПІДВИЩЕННЯ ПРОПУСКНОЇ ЗДАТНОСТІ ТА ЗМЕНШЕННЯ ЗАТРИМКИ

Розглянемо оптимізацію ефективності та продуктивності системи на кількох ієрархічних рівнях, починаючи з найнижчого — апаратно-драйверного. Налаштування цього рівня є базовими та можуть бути застосовані практично до будь-якого високонавантаженого сервера незалежно від його прикладного призначення. Оптимізація роботи мережевих адаптерів, драйверів введення-виведення та механізмів обробки переривань безпосередньо впливає на стабільність потокової передачі даних і зменшення втрат пакетів, що є критично важливим для сервісів передавання відео.

Далі увага приділяється рівню операційної системи, зокрема ядру Linux та реалізації TCP/IP-стеку. На цьому етапі можливе тонке налаштування параметрів керування чергами, вікнами прийому і передавання, механізмами повторної передачі та алгоритмами керування перевантаженнями. Такі оптимізації особливо актуальні для систем, що активно використовують TCP-з’єднання і працюють в умовах обмеженої пропускної здатності. Зменшення затримок і коливань швидкості передачі на цьому рівні створює більш сприятливі умови для подальшої обробки даних на прикладному рівні, зокрема для алгоритмів відновлення відеопотоків.

На завершальному етапі розглядаються налаштування на рівні бібліотек і прикладних програм. Більшість із них є універсальними для широкого класу веб-серверів і, зокрема, для nginx. Оптимізація буферизації, асинхронної обробки запитів та керування потоками введення-виведення дозволяє ефективніше використовувати доступні ресурси та зменшувати затримки доставки відеоданих кінцевому користувачу.

У кожній з перелічених областей оптимізації доцільно аналізувати компроміси між затримкою та пропускною здатністю. Збільшення буферів і агресивні алгоритми агрегації пакетів можуть підвищити загальну пропускну здатність, але водночас призвести до зростання затримок, що критично для потокового відео в реальному часі. Тому важливо доповнювати системні

оптимізації належним моніторингом та адаптивним підбором параметрів залежно від характеру навантаження.

У контексті відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю ці оптимізації відіграють допоміжну, але принципово важливу роль. Зниження кількості втрат і стабілізація мережевих характеристик зменшують обсяг інформації, який необхідно реконструювати на прикладному рівні. У поєднанні з алгоритмами деконволюції, що відновлюють втрачені або спотворені фрагменти відео на основі просторово-часового контексту, системні та мережеві оптимізації дозволяють досягти більш високої якості відтворення відеопотоків навіть за жорстких обмежень пропускної здатності.

2.1 Обладнання та основні характеристики

Обладнання

Процесор

Вибір процесора є одним із ключових чинників, що визначають загальну продуктивність системи, особливо в задачах, пов’язаних із криптографією, мережевою обробкою та високонавантаженими сервісами. Для забезпечення високої швидкодії асиметричних криптографічних алгоритмів, таких як RSA та ECC, доцільно обирати процесори щонайменше з підтримкою набору інструкцій AVX2 (наявність можна перевірити через avx2 у /proc/cpuinfo). Додатковою перевагою є підтримка інструкцій для ефективної роботи з великими цілими числами, зокрема BMI та ADX, які суттєво прискорюють операції множення й додавання з переносом.

Для симетричного шифрування важливо звернути увагу на апаратні оптимізації, орієнтовані на конкретні алгоритми. Так, підтримка AES-NI забезпечує значний приріст продуктивності при використанні AES-шифрів, тоді як AVX-512 є особливо корисним для сучасних алгоритмів типу ChaCha20 + Poly1305. Компанія Intel, зокрема, наводить порівняльні тести продуктивності різних поколінь

процесорів з бібліотекою OpenSSL 1.0.2, де наочно демонструється вплив цих апаратних розширень на швидкість криптографічних операцій.

Для задач, чутливих до затримок, таких як маршрутизація або обробка мережевих пакетів у реальному часі, рекомендовано зменшувати кількість NUMA-вузлів і вимикати Hyper-Threading. Це дозволяє знизити накладні витрати на синхронізацію та доступ до пам’яті. Натомість для задач, орієнтованих на високу пропускну здатність, ефективнішим є використання більшої кількості фізичних ядер разом із Hyper-Threading (за умови, що немає жорсткої прив’язки до кешу). У таких сценаріях архітектура NUMA, як правило, не є критичним фактором.

Якщо розглядати процесори Intel, доцільно звертати увагу на покоління з архітектурою Haswell або Broadwell, а ще краще — Skylake і новіші. У лінійці AMD дуже високі показники продуктивності демонструють серверні процесори EPYC, які поєднують велику кількість ядер, високу пропускну здатність пам’яті та хорошу масштабованість.

Мережева карта

Для сучасних високонавантажених серверів мінімально рекомендованою є мережева карта з пропускною здатністю 10 Гбіт/с, однак у більшості випадків доцільніше одразу орієнтуватися на 25 Гбіт/с. Якщо планується обробка великого обсягу зашифрованого трафіку (наприклад, TLS) на одному сервері, стандартних налаштувань може виявитися недостатньо. У таких випадках інколи доводиться переносити частину TLS-фреймінгу на рівень ядра операційної системи, що реалізується, наприклад, у FreeBSD або Linux.

На програмному рівні важливо обирати мережеві адаптери з якісними open-source-драйверами, які мають активні спільноти та списки розсилки. Це критично важливо для подальшого супроводу системи, оскільки проблеми з драйверами практично неминучі на певному етапі експлуатації, і наявність спільноти значно спрощує їх розв’язання.

Оперативна пам’ять

Існує просте емпіричне правило: задачі, чутливі до затримок, потребують швидшої пам’яті з меншими латентностями, тоді як задачі, орієнтовані на високу

пропускну здатність, виграють від більшого обсягу оперативної пам’яті. У серверних конфігураціях важливо також враховувати баланс між кількістю каналів пам’яті, їх частотою та архітектурою NUMA, оскільки це безпосередньо впливає на масштабованість і стабільність роботи системи під навантаженням.

Накопичувачі

Вибір дискової підсистеми значною мірою залежить від вимог до буферизації та кешування. Якщо система активно використовує буфери або кеші, оптимальним вибором є SSD-накопичувачі, які забезпечують низькі затримки доступу та високу швидкість випадкових операцій введення-виведення. У деяких випадках застосовують файлові системи, спеціально оптимізовані під флеш-пам’ять (зазвичай лог-структуровані), однак на практиці вони не завжди демонструють кращу продуктивність порівняно з класичними ext4 або xfs.

Незалежно від обраної конфігурації, важливо правильно експлуатувати флеш-накопичувачі. Зокрема, необхідно вмикати підтримку TRIM і своєчасно оновлювати мікропрограму дисків, оскільки ігнорування цих аспектів може призвести до швидкої деградації продуктивності та скорочення терміну служби накопичувачів.

Операційні системи: низькорівневий аспект

Прошивки

Одним із базових, але часто недооцінених чинників стабільної та прогнозованої роботи системи є актуальний стан прошивок. Рекомендується використовувати достатньо свіжі версії прошивок для процесора, материнської плати, мережевих адаптерів і SSD-накопичувачів, оскільки саме на цьому рівні часто усуваються критичні помилки, пов’язані з продуктивністю, енергоспоживанням і стабільністю роботи обладнання. Водночас не обов’язково завжди переходити на найновіші релізи одразу після їх виходу. На практиці більш безпечним підходом є використання передостанніх версій, за умови що в них відсутні критичні дефекти, які виправлені лише в останніх оновленнях. Така стратегія дозволяє зменшити ризик появи нових, ще не виявлених проблем, і

водночас уникнути тривалого та складного процесу пошуку причин збоїв, що можуть виникати через застарілі прошивки.

Драйвери

Подібний підхід доцільно застосовувати і при роботі з драйверами. Оптимальним варіантом є використання актуальних, але не крайніх версій драйверів, особливо у виробничих середовищах. Важливо також розділяти оновлення ядра операційної системи та оновлення драйверів, щоб мати змогу чітко ідентифікувати джерело можливих проблем. Практичним рішенням є використання механізмів на кшталт DKMS для пакування драйверів або попередня компіляція драйверних модулів під усі версії ядра, які застосовуються в системі. Завдяки цьому у випадку некоректної роботи після оновлення ядра значно легше локалізувати проблему та визначити, чи пов’язана вона безпосередньо з ядром, чи з конкретним драйвером.

Процесор

Для ефективної оптимізації роботи процесора ключовим інструментом є репозиторій ядра Linux та утиліти, що постачаються разом із ним. У дистрибутивах Ubuntu та Debian, наприклад, можна встановити пакет linux-tools, який містить набір корисних засобів для аналізу та керування поведінкою процесора. У практичних сценаріях особливо корисними є утиліти cpupower, turbostat та x86_energy_perf_policy, які дозволяють контролювати частотні режими, стан Turbo Boost, енергетичні профілі та фактичні характеристики навантаження.

Для перевірки ефективності внесених оптимізацій доцільно проводити навантажувальне тестування прикладного програмного забезпечення. Це можна реалізувати за допомогою будь-якого зручного генератора навантаження. Стрес-тести дозволяють оцінити, як система поводиться під піковими навантаженнями, і чи не виникають вузькі місця на рівні процесора. Корисним доповненням у цьому контексті є матеріали та презентації, підготовлені розробниками nginx, зокрема щодо кращих практик тестування продуктивності, які допомагають системно підійти до аналізу та оптимізації серверних рішень.

Cpupower

```
$ cpupower frequency-info
...
  driver: intel_pstate
  ...
  available cpufreq governors: performance powersave
  ...
  The governor "performance" may decide which speed to use
  ...
  boost state support:
    Supported: yes
    Active: yes
```

Переконайтеся, що в системі активовано режим Turbo Boost, а у випадку використання процесора Intel — що керування частотами здійснюється через драйвер intel_pstate, а не через acpi-cpufreq чи pcc-cpufreq. Якщо система все ще працює з acpi-cpufreq, доцільно оновити ядро операційної системи. Якщо ж оновлення ядра з певних причин неможливе, рекомендується встановити режим performance, щоб забезпечити стабільну та передбачувану продуктивність.

При використанні intel_pstate навіть режим powersave, як правило, забезпечує достатньо високий рівень продуктивності завдяки динамічному масштабуванню частот. Однак фактична ефективність цього режиму залежить від конкретного навантаження та конфігурації системи, тому її доцільно перевірити експериментально у власному робочому середовищі.

Що стосується режимів простою, то для детального аналізу реальної поведінки процесора доцільно скористатися утилітою turbostat. Вона дозволяє безпосередньо звертатися до процесорних регістрів MSR і отримувати актуальну інформацію про енергоспоживання, робочі частоти, а також так звані стани простою (Idle States), у яких перебуває процесор під час роботи системи.

```
# turbostat --debug -P
```

... Avg_MHz Busy% ... CPU%c1 CPU%c3 CPU%c6 ... Pkg%pc2 Pkg%pc3 Pkg%pc6 ...

У цьому випадку можна спостерігати фактичну робочу частоту процесора, яка відображає його реальний стан, на відміну від даних з /proc/cpuinfo, що часто показують лише номінальні або усереднені значення і не завжди відповідають дійсній поведінці системи. Крім того, доступна інформація про поточний стан окремого ядра або групи ядер, що дозволяє точніше оцінити, як процесор використовує свої ресурси під навантаженням і в режимах простою.

Якщо навіть за умови використання драйвера intel_pstate виявляється, що процесор проводить у станах простою більше часу, ніж очікувалося, існує кілька можливих шляхів оптимізації. По-перше, можна переключити регулятор керування частотами у режим performance, що змусить процесор утримувати вищі частоти та швидше реагувати на зростання навантаження. По-друге, для додаткового підвищення продуктивності доцільно налаштувати політику енергоспоживання за допомогою утиліти x86_energy_perf_policy, обравши параметри, орієнтовані на максимальну швидкодію, а не на економію енергії.

Для задач, які є особливо чутливими до затримок, наприклад у мережевій обробці або реальному часі, можливі більш радикальні заходи оптимізації. Зокрема, можна скористатися інтерфейсом /dev/cpu_dma_latency, який дозволяє обмежити глибину станів простою процесора і зменшити час пробудження ядер. Крім того, для UDP-трафіку іноді застосовують режим busy-polling, який зменшує затримки обробки пакетів за рахунок активного опитування, хоча й збільшує навантаження на процесор.

Прив’язка до процесора

Ще одним ефективним способом зменшення затримок є жорстка прив’язка потоку або процесу до конкретного ядра процесора. Такий підхід дозволяє більш детерміновано керувати виконанням завдань і зменшувати накладні витрати, пов’язані з плануванням. Наприклад, у веб-сервері nginx передбачена директива worker_cpu_affinity, яка автоматично закріплює кожен робочий процес за

визначеним ядром CPU. У результаті усувається міграція процесів або потоків між ядрами, що позитивно впливає на локальність даних у кеші.

Фіксація процесів на ядрах дозволяє зменшити кількість промахів кешу та збоїв сторінок пам’яті, оскільки дані й інструкції залишаються у кешах одного й того самого ядра. Крім того, у багатьох випадках спостерігається незначне зростання кількості інструкцій, виконаних за один такт, що загалом підвищує ефективність виконання коду. Перевірити та кількісно оцінити такі покращення можна за допомогою інструментів профілювання, зокрема perf stat, який надає детальну статистику щодо виконання інструкцій, кеш-промахів і інших апаратних подій.

Водночас процесорна прив’язка має і зворотний бік. Жорстке закріплення процесів за ядрами може негативно впливати на загальну пропускну здатність системи, оскільки процесам доводиться довше чекати на звільнення саме «свого» ядра, навіть якщо інші ядра простоюють. У таких умовах зростає час очікування в черзі планувальника, що особливо помітно при нерівномірному або змінному навантаженні.

Для аналізу цього ефекту доцільно використовувати інструменти трасування та моніторингу черг виконання. Зокрема, можна запустити runqlat для одного з PID робочих процесів nginx і відстежити розподіл часу очікування в черзі виконання. Це дозволяє об’єктивно оцінити, чи виправдана процесорна прив’язка в конкретному сценарії, та підібрати оптимальний баланс між мінімізацією затримок і максимальною продуктивністю системи.

```
usecs               : count     distribution
    0 -> 1          : 819      |                                        |
    2 -> 3          : 58888    |****************************            |
    4 -> 7          : 77984    |****************************************|
    8 -> 15         : 10529    |*****                                   |
   16 -> 31         : 4853     |**                                      |
   ...
 4096 -> 8191       : 34       |                                        |
 8192 -> 16383      : 39       |                                        |
16384 -> 32767      : 17       |                                        |
```

Рис. 2.1 Відстежування за допомогою запуску runqlat на одному з PID nginx-воркера

Пам’ять

Усі налаштування, пов’язані з керуванням пам’яттю (Memory Management), значною мірою залежать від характеру конкретного робочого навантаження та поведінки прикладного процесу. Тому універсальних рішень у цій сфері не існує, і більшість параметрів потребують експериментального підбору. Водночас можна сформулювати кілька загальних рекомендацій, які доцільно розглядати як відправну точку для подальшої оптимізації:

- **налаштуйте Transparent Huge Pages у режим madvise** і вмикайте їх лише тоді, коли є впевненість у їх реальній користі; в іншому разі прагнення, наприклад, до зменшення затримки на 20 % може призвести до суттєвого погіршення загальної продуктивності системи;
- **якщо використовується лише один NUMA-вузол**, встановіть параметр vm.zone_reclaim_mode у значення 0, щоб уникнути зайвих спроб ядра повторно використовувати локальні зони пам’яті ціною зростання затримок.

Сучасні процесори фактично є набором кількох обчислювальних модулів, з’єднаних між собою високошвидкісною шиною та спільно використовуваних ресурсів. Це охоплює широкий спектр рівнів — від кешу L1, спільного для HT-ядер, до кешу L3, а також оперативної пам’яті, PCIe-з’єднань і інших компонентів у межах одного сокета. Саме така архітектура і лежить в основі концепції NUMA (Non-Uniform Memory Access), яка передбачає наявність кількох зон виконання та зберігання даних із різними характеристиками доступу, об’єднаних швидкісним інтерконектом.

Детальний і всебічний розбір архітектури NUMA, принципів її роботи та практичних сценаріїв застосування наведено у серії матеріалів Френка Деннемана NUMA Deep Dive Series, яка вважається одним із базових джерел з цієї тематики.

У практичних умовах можливі кілька стратегій роботи з NUMA, кожна з яких має свої переваги та недоліки:

- **ігнорувати NUMA**, вимкнувши її в BIOS або запускаючи програмне забезпечення з параметром numactl --interleave=all, що забезпечує посередню, але відносно стабільну продуктивність;

- **відмовитися від NUMA на апаратному рівні**, використовуючи одноузлові сервери, як це реалізовано, наприклад, у платформі OCP Yosemite, яку застосовує Facebook;
- **прийняти NUMA як даність** і оптимізувати розміщення процесів та пам’яті як у просторі ядра, так і в користувацькому просторі, досягаючи максимальної ефективності за рахунок усвідомленого керування локальністю даних.

Вибір конкретного підходу має ґрунтуватися на типі навантаження, вимогах до затримок і пропускної здатності, а також на доступних ресурсах і складності адміністрування системи.

Розглянемо третій підхід, оскільки у двох попередніх варіантах обсяг необхідних оптимізацій є мінімальним або взагалі незначним.

Для коректного та ефективного використання архітектури NUMA доцільно сприймати кожен NUMA-вузол як окремий логічний сервер із власними обчислювальними та пам’яттєвими ресурсами. Насамперед необхідно проаналізувати топологію системи, що можна зробити за допомогою команди numactl --hardware, яка надає інформацію про кількість NUMA-вузлів, закріплені за ними ядра процесора та області оперативної пам’яті.

```
$ numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 16 17 18 19
node 0 size: 32149 MB
node 1 cpus: 4 5 6 7 20 21 22 23
node 1 size: 32213 MB
node 2 cpus: 8 9 10 11 24 25 26 27
node 2 size: 0 MB
node 3 cpus: 12 13 14 15 28 29 30 31
node 3 size: 0 MB
node distances:
node   0   1   2   3
  0:  10  16  16  16
  1:  16  10  16  16
  2:  16  16  10  16
  3:  16  16  16  10
```

Рис. 2.2 Перевірка топології за допомогою numactl --hardware

Необхідно звернути увагу на такі параметри:

- кількість NUMA-вузлів у системі;
- обсяг оперативної пам’яті, доступної для кожного вузла;
- кількість процесорних ядер, закріплених за кожним вузлом;
- відстань між вузлами, яка відображає відносну вартість доступу до пам’яті інших вузлів.

Наведений приклад є невдалим, оскільки система містить чотири NUMA-вузли, при цьому деякі з них не мають прив’язаної пам’яті. За такої конфігурації неможливо розглядати кожен вузол як окремий логічний сервер без суттєвих втрат — зокрема, це призводить до фактичного невикористання половини процесорних ядер.

Це можна перевірити за допомогою numastat:

```
$ numastat -n -c
                 Node 0   Node 1 Node 2 Node 3    Total
               -------- -------- ------ ------ --------
Numa_Hit        26833500 11885723      0      0 38719223
Numa_Miss          18672  8561876      0      0  8580548
Numa_Foreign     8561876    18672      0      0  8580548
Interleave_Hit    392066   553771      0      0   945836
Local_Node       8222745 11507968      0      0 19730712
Other_Node      18629427  8939632      0      0 27569060
```

Рис. 2.3 Перевірка за допомогою numastat

Також за допомогою numastat можна отримати статистику використання пам'яті кожного вузла у форматі /proc/meminfo:

```
$ numastat -m -c
                 Node 0 Node 1 Node 2 Node 3 Total
                 ------ ------ ------ ------ -----
MemTotal          32150  32214      0      0 64363
MemFree             462   5793      0      0  6255
MemUsed           31688  26421      0      0 58109
Active            16021   8588      0      0 24608
Inactive          13436  16121      0      0 29557
Active(anon)       1193    970      0      0  2163
Inactive(anon)      121    108      0      0   229
Active(file)      14828   7618      0      0 22446
Inactive(file)    13315  16013      0      0 29327
...
FilePages         28498  23957      0      0 52454
Mapped              131    130      0      0   261
AnonPages           962    757      0      0  1718
Shmem               355    323      0      0   678
KernelStack          10      5      0      0    16
```

Рис. 2.4 Статистика використання пам’яті

Оскільки NUMA-вузли здебільшого є симетричними за своїми характеристиками, можна закріпити окремий екземпляр прикладного застосунку за кожним вузлом, використовуючи команду numactl --cpunodebind=X --membind=X, а після цього запустити його на окремому мережевому порту. Такий підхід дозволяє підвищити загальну пропускну здатність системи за рахунок паралельного використання кількох вузлів, а також зменшити затримки завдяки збереженню локальності доступу до пам’яті.

Оцінити ефективність такого розміщення з урахуванням NUMA-архітектури можна шляхом вимірювання затримок операцій доступу до пам’яті. Наприклад, за допомогою інструмента funclatency з набору BCC можна заміряти латентність операцій, що інтенсивно використовують пам’ять, таких як memmove. Крім того, спостерігати за впливом NUMA-оптимізацій на рівні ядра доцільно за допомогою perf stat, відстежуючи відповідні події, пов’язані з підсистемою пам’яті та роботою планувальника.

Остання група оптимізацій, пов’язаних з NUMA, для мережевих навантажень із інтенсивним використанням мережі зумовлена тим, що мережева карта є пристроєм PCIe, а кожен такий пристрій фізично прив’язаний до конкретного

NUMA-вузла. Унаслідок цього для окремих процесорів затримка доступу до мережевих ресурсів буде меншою, ніж для інших, залежно від їхньої топологічної близькості до відповідного пристрою.

Можливі способи оптимізації цієї взаємодії будуть розглянуті в окремому розділі, присвяченому прив’язці мережевої карти до процесора. На даному етапі доцільно перейти до розгляду особливостей шини PCI Express та її впливу на продуктивність системи.

PCIe

Зазвичай немає потреби глибоко занурюватися в аналіз і усунення проблем, пов’язаних із PCI Express, якщо в системі не спостерігаються явні апаратні збої. Водночас доцільно хоча б на базовому рівні контролювати ключові параметри для всіх PCIe-пристроїв, зокрема ширину шини, швидкість з’єднання, а також наявність попереджень типу RxErr або BadTLP. Такий попередній моніторинг дозволяє заощадити значну кількість часу, який інакше довелося б витратити на пошук причин проблем, пов’язаних із несправним обладнанням або некоректним узгодженням PCIe-з’єднань.

Для отримання цієї інформації зручно використовувати утиліту lspci, яка надає детальні відомості про параметри та стан PCIe-пристроїв у системі.

```
# lspci -s 0a:00.0 -vvv
...
LnkCap: Port #0, Speed 8GT/s, Width x8, ASPM L1, Exit Latency L0s <2us, L1 <16us
LnkSta: Speed 8GT/s, Width x8, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
...
Capabilities: [100 v2] Advanced Error Reporting
UESta:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- ...
UEMsk:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- ...
UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- ...
CESta:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr-
CEMsk:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
```

Рис. 2.5 Попередній моніторинг за допомогою утиліти lspci

Регулювання пакетів

Усі сучасні мережеві адаптери спроєктовані з урахуванням роботи в багатопроцесорних і багатоядерних системах. З цієї причини вони підтримують

розподіл мережевих пакетів між кількома віртуальними чергами, зазвичай у кількості, що відповідає числу процесорних ядер. Коли такий розподіл виконується безпосередньо на апаратному рівні, він називається RSS (Receive Side Scaling). Якщо ж балансування пакетів між процесорами здійснюється засобами операційної системи, цей механізм має назву RPS (Receive Packet Steering), а його передавальний аналог позначається як XPS (Transmit Packet Steering).

Існує також механізм RFS (Receive Flow Steering), за якого операційна система намагається спрямовувати пакети до тих процесорів, які вже обробляють відповідні мережеві сокети. У випадку, коли подібне перенаправлення реалізується на рівні апаратного забезпечення, його називають прискореним RFS або aRFS. Кожен із цих підходів має власні особливості та по-різному впливає на затримки й пропускну здатність системи.

Нижче наведено кілька практичних рекомендацій щодо ефективного використання зазначених механізмів:

- якщо використовується сучасне мережеве обладнання з пропускною здатністю 25 Гбіт/с і більше, воно, як правило, має достатню кількість апаратних черг і велику таблицю непрямої адресації (indirection table), що дозволяє застосовувати RSS для розподілу навантаження між усіма ядрами процесора; варто враховувати, що деякі старі моделі мережевих карт можуть підтримувати RSS лише для перших 16 процесорів;
- можна спробувати ввімкнути RPS у таких випадках:
    1. якщо кількість процесорних ядер перевищує кількість апаратних черг мережевої карти і є готовність пожертвувати затримкою заради збільшення загальної пропускної здатності;
    2. якщо використовується внутрішнє тунелювання (наприклад, GRE або IP-in-IP), за якого мережева карта не має змоги коректно застосовувати RSS;
- не рекомендується вмикати RPS на відносно старих процесорах, які не підтримують режим x2APIC, оскільки це може призвести до додаткових накладних витрат і погіршення продуктивності;

- прив’язка кожного процесорного ядра до власної черги передавання за допомогою XPS у більшості випадків є доцільним і ефективним рішенням;
- ефективність використання RFS значною мірою визначається характером робочого навантаження, а також тим, чи застосовується процесорна прив’язка потоків або процесів.

Коректне поєднання цих механізмів дозволяє досягти балансу між мінімізацією затримок і максимальною пропускною здатністю мережевої підсистеми, що є особливо важливим для високонавантажених серверних середовищ.

Flow Director та ATR

Увімкнений механізм Flow Director (або fdir у термінології Intel) за замовчуванням працює в режимі Application Targeting Routing (ATR). У цьому режимі реалізується прискорений RFS (aRFS), який ґрунтується на вибірковому аналізі (семплюванні) мережевих пакетів і подальшому спрямуванні потоків до того процесорного ядра, на якому ці пакети, з найбільшою ймовірністю, уже обробляються. Такий підхід дозволяє підвищити локальність даних і зменшити накладні витрати, пов’язані з міграцією потоків між ядрами.

Статистичні дані щодо роботи Flow Director можна отримати за допомогою утиліти ethtool, переглядаючи відповідні лічильники, наприклад через ethtool -S eth0 з фільтрацією показників, що містять fdir. Це дозволяє оцінити, як активно використовується механізм і чи відбувається очищення або переналаштування таблиць потоків.

Хоча компанія Intel заявляє, що Flow Director у певних сценаріях здатен підвищувати продуктивність, результати окремих досліджень свідчать про можливі побічні ефекти. Зокрема, виявлено, що приблизно до 1 % пакетів може надходити з порушенням початкового порядку. Для TCP-з’єднань це є критичним фактором, оскільки навіть незначне переупорядкування пакетів може негативно впливати на пропускну здатність і стабільність з’єднання. Тому доцільно самостійно протестувати роботу Flow Director у власному середовищі та оцінити його

корисність для конкретного типу навантаження, зокрема відстежуючи лічильник TCPOFOQueue, який відображає кількість пакетів, що надійшли поза порядком.

Операційні системи: мережевий стек

Існує величезна кількість книг, відеоматеріалів і практичних посібників, присвячених налаштуванню мережевого стеку Linux. У багатьох із них досі тиражується так званий «карго-культ sysctl.conf», коли великі набори параметрів бездумно копіюються з конфігурацій, актуальних для ядер десятирічної давнини. Хоча сучасні версії ядра Linux уже не потребують такого агресивного ручного налаштування, а більшість нових TCP/IP-механізмів увімкнені та коректно налаштовані за замовчуванням, практика використання застарілих sysctl.conf, характерних для ядер 2.6.18 або 2.6.32, усе ще залишається поширеною.

Замість цього доцільно зосередитися на вимірюванні та аналізі фактичної ефективності мережевих оптимізацій. Для цього можна виконати такі кроки:

- зібрати системні TCP-метрики за допомогою файлів /proc/net/snmp та /proc/net/netstat;
- додати метрики активних з’єднань, використовуючи ss -n --extended --info або виклики getsockopt(TCP_INFO) і getsockopt(TCP_CC_INFO) безпосередньо в серверному коді;
- зібрати зразки TCP-потоків за допомогою інструмента tcptrace;
- проаналізувати RUM-метрики (Real User Monitoring) на рівні застосунку або браузера кінцевого користувача.

Як джерела практичних знань щодо мережевих оптимізацій особливо корисними є виступи та матеріали спеціалістів з CDN, оскільки вони, як правило, працюють у середовищах з екстремальними вимогами до продуктивності та затримок. Прикладом можуть слугувати доповіді Fastly на LinuxCon Australia. Також варто звертати увагу на презентації та обговорення розробників ядра Linux на конференціях NetDevConf і Netconf, де розглядаються найновіші зміни та майбутні напрями розвитку мережевого стеку.

Окремо слід відзначити ґрунтовні матеріали від PackageCloud, присвячені мережевому стеку Linux, у яких акцент зроблено не на «сліпому» тюнінгу, а на системному моніторингу та аналізі:

- *Monitoring and Tuning the Linux Networking Stack: Receiving Data*;
- *Monitoring and Tuning the Linux Networking Stack: Sending Data*.

На завершення варто наголосити на простій, але вкрай важливій рекомендації: оновлюйте ядро операційної системи. Сучасні версії ядра містять велику кількість мережевих удосконалень, і мова йде не лише про відносно старі ініціативи на кшталт IW10, а й про такі механізми, як автоматичний вибір розміру TSO, FQ, pacing, TLP та RACK. Додатковим бонусом від оновлення є значні покращення масштабованості, зокрема усунення кешу маршрутизації, поява неблокуючих сокетів прослуховування, підтримка SO_REUSEPORT та багато інших оптимізацій, які суттєво спрощують побудову високопродуктивних мережевих сервісів.

2.2 Огляд відновлювальних алгоритмів

Серед відносно недавніх матеріалів, присвячених роботі з мережею в Linux, особливо вирізняється документ Making Linux TCP Fast. У ньому на кількох сторінках у стислій формі зібрано та систематизовано покращення, які накопичувалися в ядрі операційної системи протягом багатьох років. У цьому документі TCP-стек на стороні відправника представлено як набір окремих функціональних компонентів, кожен з яких відповідає за певний етап обробки та передавання даних.

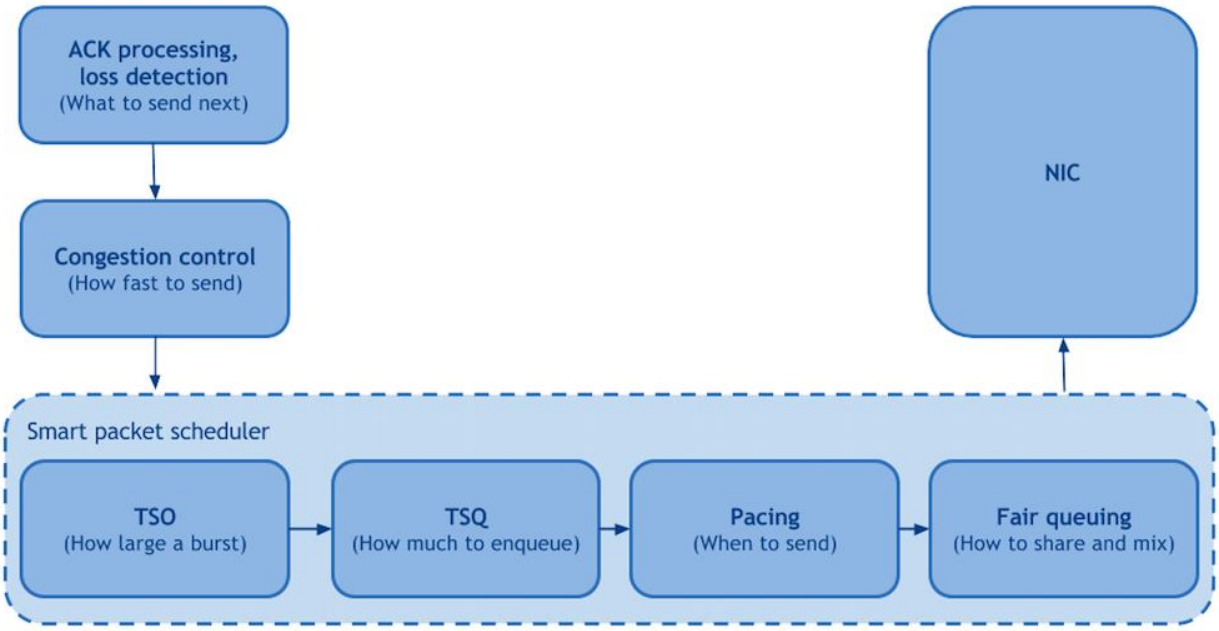


Рис. 2.6 Функціональна схема TCP-стеку на стороні відправника в Linux, що ілюструє послідовність обробки ACK і виявлення втрат, керування перевантаженням, інтелектуальне планування пакетів (TSO, TSQ, pacing, fair queuing) та передачу даних на мережевий адаптер (NIC)

Fair queueing та pacing

Механізм fair queueing відповідає за дотримання «справедливості» під час обслуговування кількох TCP-потоків, запобігаючи ситуаціям, коли один потік блокує чергу і витісняє інші. Завдяки цьому зменшується взаємна конкуренція між потоками, скорочується ймовірність переповнення черг і, як наслідок, знижується частота відкидання пакетів. Такий підхід особливо важливий у багатопотокових сценаріях, де одночасно передається велика кількість даних різного типу, зокрема мультимедійний трафік.

Механізм pacing, у свою чергу, забезпечує рівномірний розподіл пакетів у часі відповідно до швидкості, визначеної алгоритмом керування перевантаженнями (Congestion Control). Замість передачі пакетів «ривками» pacing формує більш гладкий потік, що додатково зменшує пікові навантаження на мережу. Це приводить до подальшого скорочення втрат пакетів і, відповідно, до підвищення ефективної пропускної здатності каналу.

У Linux обидва механізми — fair queueing і pacing — реалізовані за допомогою дисципліни черг fq qdisc. Спочатку ці можливості були обов’язковими для коректної роботи алгоритму BBR, однак з часом вони довели свою корисність і в поєднанні з класичними алгоритмами керування перевантаженнями, такими як CUBIC. На практиці їх застосування дозволяє досягти зниження втрат пакетів на 15–20 %, що безпосередньо трансформується у зростання пропускної здатності для алгоритмів, заснованих на втраті пакетів (loss-based CC).

Водночас слід враховувати обмеження сумісності: використання fair queueing і pacing на застарілих версіях ядра Linux (раніше 3.19) не рекомендується, оскільки в таких умовах ці механізми можуть впливати на звичайні ACK-пакети, що призводить до деградації аплоаду та проблем у RPC-викликах.

У мережах з обмеженою пропускною здатністю ці механізми мають додаткову цінність. Зменшення втрат і стабілізація трафіку на транспортному рівні знижують обсяг пошкоджених або втрачених фрагментів відео. Це, у свою чергу, полегшує роботу алгоритмів деконволюції, які застосовуються на прикладному рівні для реконструкції відеопотоків. За умов більш рівномірної доставки даних деконволюційні моделі можуть ефективніше відновлювати просторово-часову структуру відео, спираючись на контекст сусідніх кадрів, що дозволяє досягати вищої якості відтворення навіть за наявності обмежень пропускної здатності мережі.

Автоматичний вибір розміру TSO та TSQ

Механізми автоматичного вибору розміру TSO (TCP Segmentation Offload) і TSQ (TCP Small Queues) відіграють важливу роль у сучасному TCP-стеку Linux. Обидві функції спрямовані на обмеження надмірної буферизації всередині транспортного рівня, що безпосередньо впливає на затримки передавання даних. Завдяки динамічному підбору оптимальних розмірів сегментів і черг вдається зменшити латентність без негативного впливу на загальну пропускну здатність каналу. Це особливо важливо для потокових і інтерактивних сервісів, де стабільність затримок має не менше значення, ніж максимальна швидкість передавання.

Керування перевантаженнями

Алгоритми керування перевантаженнями (Congestion Control, CC) самі по собі є складною та об’ємною темою, яка в останні роки активно обговорюється як у наукових колах, так і в інженерних спільнотах. Частина цих напрацювань уже реалізована на практиці у вигляді конкретних алгоритмів, зокрема tcp_cdg (розробка CAIA), tcp_nv (Facebook) та tcp_bbr (Google). Не заглиблюючись у внутрішню реалізацію кожного з них, варто зазначити спільну рису: у всіх цих алгоритмах індикація перевантаження ґрунтується переважно на зростанні затримок (delay), а не лише на факті втрати пакетів.

Серед сучасних підходів особливо вирізняється алгоритм BBR, який є одним із найбільш задокументованих, ретельно протестованих і практично застосовних рішень у сфері керування перевантаженнями. BBR будує модель мережевого шляху на основі частки успішно доставлених пакетів, оцінюючи доступну пропускну здатність і мінімальний час обігу (RTT). Далі на основі цієї моделі виконуються керуючі цикли, спрямовані на одночасне збільшення ефективної ширини пропускання та мінімізацію затримок. Саме така поведінка є особливо цінною для проксі-стеків і систем доставки контенту, де важливо швидко адаптуватися до змін мережевих умов.

Попередні результати експериментів із використанням BBR на прикордонних вузлах (Edge PoP) показали помітне зростання швидкості завантаження файлів. Це підтверджує доцільність застосування алгоритмів, орієнтованих на затримку, у реальних мережевих середовищах із змінною пропускною здатністю та неоднорідними умовами передавання.

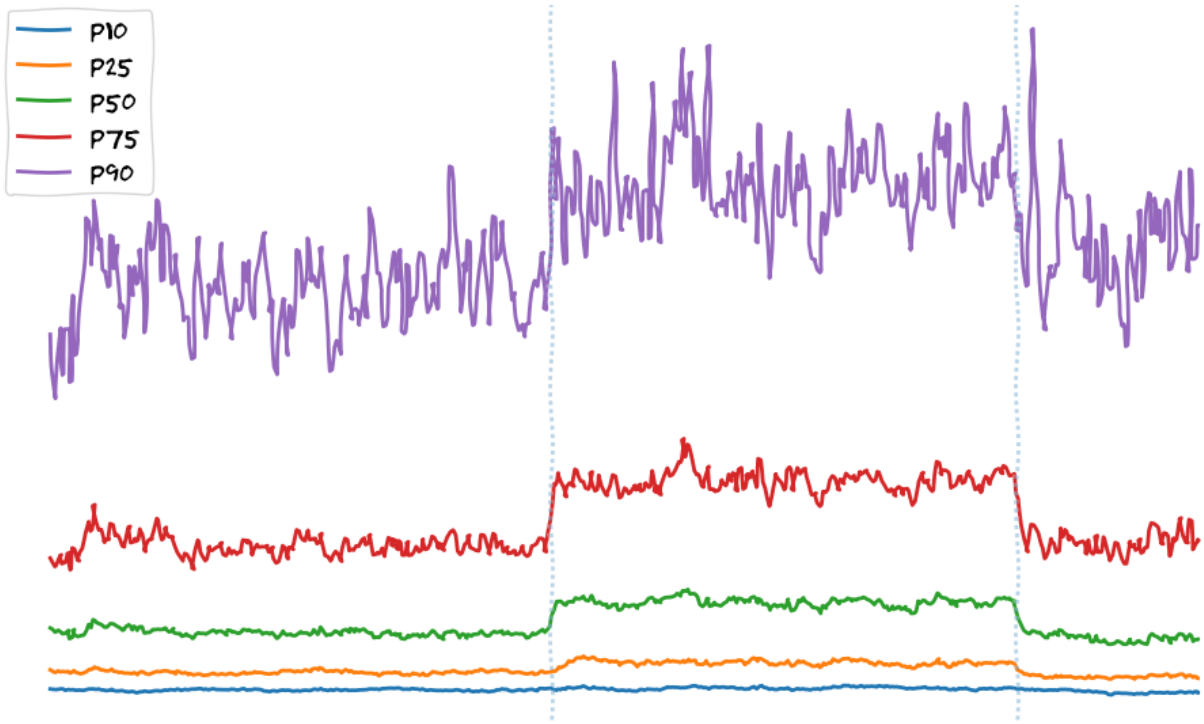


Рис. 2.7 Шестигодинний експеримент із TCP BBR у Tokyo PoP: вісь x — час, вісь y — швидкість завантаження на клієнта

Збільшення швидкості передачі даних спостерігалося по всіх перцентилях користувачів, а не лише в окремих сегментах. Це є показовим результатом, оскільки при зміні або оптимізації бекенда подібного ефекту, як правило, не виникає. У таких випадках покращення зазвичай помітне лише для користувачів із високими значеннями перцентилів, наприклад p90 і вище, тобто для тих, хто має найшвидше інтернет-з’єднання. Для решти користувачів традиційно вважається, що їхня швидкість уже обмежена пропускною здатністю каналу доступу.

Натомість результати налаштувань на мережевому рівні, зокрема зміна алгоритмів керування перевантаженнями або ввімкнення механізмів fair queueing і pacing, демонструють іншу картину. Вони свідчать про те, що для значної частини користувачів вузьким місцем є не стільки сама пропускна здатність каналу, скільки обмеження, зумовлені поведінкою TCP. Іншими словами, можна говорити про наявність так званої «TCP-обмеженості», коли ефективна швидкість передачі визначається не фізичними характеристиками мережі, а особливостями роботи транспортного протоколу.

Для тих, хто бажає детальніше ознайомитися з алгоритмом BBR, варто звернути увагу на оглядові матеріали від APNIC, де доступно подано базові принципи роботи цього алгоритму та наведено його порівняння з класичними підходами до керування перевантаженнями, заснованими на втраті пакетів. Більш поглиблену та технічну інформацію можна знайти в архівах поштової розсилки bbr-dev, у якій зібрано чимало корисних обговорень і посилань на додаткові ресурси. Загалом же, для системного вивчення теми керування перевантаженнями доцільно стежити за діяльністю Internet Congestion Control Research Group, яка активно досліджує й розвиває сучасні підходи у цій галузі.

Обробка ACK та виявлення втрат пакетів

Далі розглянемо механізми виявлення втрати пакетів (loss detection) у TCP. Насамперед знову слід наголосити на важливості використання актуальної версії ядра операційної системи. TCP-стек Linux постійно еволюціонує: до нього додаються нові евристики, такі як TLP (Tail Loss Probe) та RACK (Recent ACKnowledgment), тоді як застарілі механізми на кшталт FACK або ER поступово вилучаються. Нові алгоритми зазвичай активуються за замовчуванням, тому після оновлення ядра немає потреби у додатковому ручному налаштуванні — система автоматично починає використовувати більш ефективні методи виявлення втрат і відновлення передачі.

Сучасні підходи до loss detection значно швидше реагують на втрати пакетів і зменшують час простою з’єднання, що позитивно впливає як на затримки, так і на загальну пропускну здатність. Це особливо важливо для застосунків, чутливих до латентності, зокрема для веб-сервісів і потокової передачі даних.

Пріоритизація користувацького простору та HOL-блокування

API сокетів у користувацькому просторі (userspace socket API) передбачає механізм явної буферизації. Після того як окремі чанки даних передані в TCP-стек, їх уже неможливо переупорядкувати. У сценаріях з мультиплексуванням, наприклад у HTTP/2, це може призводити до Head-of-Line (HOL) блокування та інверсії пріоритетів потоків, коли менш важливі дані затримують передачу більш пріоритетних.

Для пом’якшення цієї проблеми були розроблені спеціальні опції сокетів і відповідний параметр sysctl net.ipv4.tcp_notsent_lowat. Вони дозволяють визначити межі, в яких сокет вважається доступним для запису. Фактично це означає, що механізми сповіщення (наприклад, epoll) можуть сигналізувати про готовність до запису навіть тоді, коли фактична передача даних ще обмежена. Такий підхід здатен частково вирішити проблеми з пріоритизацією HTTP/2, однак водночас може негативно вплинути на загальну пропускну здатність. Тому доцільно перевіряти ефект від цих налаштувань експериментально, з урахуванням конкретного типу навантаження.

Sysctl-параметри

Говорячи про оптимізацію мережевої підсистеми, важко оминути тему налаштування sysctl. Проте перш ніж вносити зміни, варто чітко усвідомити, які параметри змінювати не слід:

- **net.ipv4.tcp_tw_recycle=1**: не використовуйте цей параметр — для клієнтів за NAT він уже давно створює проблеми, а в сучасних ядрах його застосування може призвести до некоректної роботи з’єднань для всіх користувачів;
- **net.ipv4.tcp_timestamps=0**: не вимикайте цю опцію, доки повністю не розумітимете всі побічні ефекти; зокрема, її вимкнення може призвести до втрати масштабування вікон і підтримки SACK у механізмах syncookie.

Натомість доцільніше зосередитися на таких налаштуваннях:

- **net.ipv4.tcp_slow_start_after_idle=0**: основна проблема повільного старту після простою полягає в тому, що «простій» визначається як один RTO, що часто є надто малим значенням;
- **net.ipv4.tcp_mtu_probing=1**: корисно в середовищах із наявністю ICMP «чорних дір» між сервером і клієнтами, які на практиці зустрічаються доволі часто;
- **net.ipv4.tcp_rmem** та **net.ipv4.tcp_wmem**: ці параметри слід налаштовувати з урахуванням BDP (Bandwidth-Delay Product), пам’ятаючи, що надмірне збільшення буферів не завжди дає позитивний ефект;

- **echo 2 > /sys/module/tcp_cubic/parameters/hystart_detect**: при використанні поєднання FQ і CUBIC це може допомогти усунути проблему надто раннього виходу tcp_cubic з фази повільного старту.

Окремо варто згадати чернетку RFC, підготовлену Даніелем Штенбергом, автором curl, під назвою *TCP Tuning for HTTP*. У цьому документі зроблено спробу зібрати та систематизувати всі системні налаштування, які можуть бути корисними для HTTP-застосунків. Хоча документ і не був доведений до фінального стандарту, він залишається цінним орієнтиром для розуміння взаємозв'язку між параметрами TCP та поведінкою сучасних веб-протоколів.

Рівень застосунку: середній рівень

Інструментарій

Так само, як і у випадку з ядром операційної системи, середовище користувацького простору потребує регулярного оновлення та актуалізації. Першим кроком до ефективного аналізу й оптимізації слід зробити оновлення інструментів діагностики та профілювання. Зокрема, доцільно використовувати свіжі версії таких утиліт, як perf, bcc та інших супутніх засобів, за потреби попередньо запакувавши їх або встановивши з перевірених репозиторіїв.

Після оновлення інструментарію можна переходити безпосередньо до налаштування та спостереження за поведінкою системи. У межах цього етапу основний акцент робиться на профілюванні процесора, зокрема за допомогою perf top, аналізі on-CPU flame-графіків, а також використанні спеціалізованих гістограм, отриманих з утиліти funclatency у складі bcc. Такий підхід дозволяє глибше зрозуміти, де саме витрачаються обчислювальні ресурси та які ділянки коду є критичними з точки зору продуктивності.

Інструментарій для компілювання

Якщо є потреба створити збірку програмного забезпечення з апаратними оптимізаціями, необхідно використовувати сучасні компілятори та супутні інструменти. Багато з них уже інтегровані в бібліотеки, які широко застосовуються у веб-серверах та серверних застосунках, і дозволяють максимально ефективно задіяти можливості сучасних процесорів.

Окрім приросту продуктивності, нові версії компіляторів пропонують розширені механізми забезпечення безпеки. До таких можливостей належать, зокрема, опції на кшталт -fstack-protector-strong або SafeStack, які підвищують стійкість програм до експлуатації вразливостей, пов’язаних із переповненням стеку. Також актуальний компіляторний інструментарій є незамінним у випадках, коли необхідно виконувати тестування за допомогою бінарних файлів, скомпільованих із використанням різноманітних санітайзерів, що дозволяють виявляти помилки пам’яті та інші дефекти ще на етапі розробки.

Рівень застосунку: високорівневі оптимізації

Для аналізу ефективності оптимізацій на високорівневому прикладному рівні необхідно збирати RUM-дані (Real User Monitoring), які відображають реальний досвід кінцевих користувачів. У браузерах для цього доцільно використовувати Navigation Timing API та Resource Timing API. Ключовими показниками при такому аналізі є TTFB (Time To First Byte) і TTV/TTI (Time To View / Time To Interactive), які безпосередньо характеризують сприйняття швидкодії веб-застосунку користувачем. Процес ітеративної оптимізації значно спрощується, якщо ці метрики зберігаються у форматах, зручних для формування запитів, агрегацій і побудови графіків.

Компресія

У веб-серверах механізм стиснення починається з файлу mime.types, який визначає відповідність між розширенням файлу та його MIME-типом. На основі цього файлу задається перелік типів даних, що підлягають стисненню, наприклад за допомогою директиви gzip_types. Якщо виникає потреба автоматизувати формування такого списку, можна скористатися базою mime-db, яка дозволяє генерувати mime.types із додатковою ознакою compressible == true для подальшого використання у gzip_types.

Увімкнення gzip-стиснення має враховувати низку важливих моментів:

- стиснення збільшує споживання оперативної пам’яті, однак цю проблему можна частково вирішити шляхом обмеження параметра gzip_buffers;

- через додаткову буферизацію зростає TTFB, але цей ефект можна зменшити, використовуючи директиву gzip_no_buffer.

Варто зазначити, що HTTP-стиснення в nginx не обмежується лише gzip. Існує сторонній модуль ngx_brotli, який забезпечує приблизно на 30 % кращий коефіцієнт стиснення порівняно з gzip, що особливо актуально для сучасних браузерів.

Статичні та динамічні дані

Підхід до налаштування стиснення доцільно розділяти для статичних і динамічних ресурсів.

Для статичних даних оптимальним є використання попереднього стиснення. У цьому випадку ресурси стискаються заздалегідь під час процесу збірки, а сервер лише віддає готові стиснені файли. Такий підхід дозволяє досягти максимального коефіцієнта стиснення без додаткового навантаження на процесор під час обробки запитів. Практичні аспекти використання gzip і brotli для статичного контенту детально описані в матеріалі *Deploying Brotli for static content*.

Для динамічних даних необхідно ретельно балансувати повний цикл обробки:
час на стиснення → час на передавання → час на розпакування на стороні клієнта.

У таких сценаріях використання максимального рівня стиснення може виявитися недоцільним не лише через збільшене навантаження на CPU, але й через негативний вплив на TTFB. Тому рівень стиснення слід підбирати експериментально з урахуванням типу контенту та характеру навантаження.

Буферизація у проксі

Буферизація всередині проксі-рівня може суттєво впливати на продуктивність веб-сервера, особливо з точки зору затримок. У проксі-модулі nginx передбачено низку параметрів, які дозволяють тонко налаштовувати поведінку буферизації залежно від того, де саме зберігаються дані та на якому етапі обробки вони перебувають. За допомогою директив proxy_request_buffering і proxy_buffering можна незалежно керувати буферизацією запитів і відповідей.

Якщо буферизацію увімкнено, верхня межа споживання пам’яті визначається параметрами client_body_buffer_size та proxy_buffers. Після досягнення цих меж дані починають тимчасово зберігатися на диску. Для відповідей таку поведінку можна вимкнути, встановивши proxy_max_temp_file_size у значення 0.

Найпоширеніші сценарії використання буферизації такі:

- буферизація запитів і відповідей до певного обсягу в оперативній пам’яті з подальшим скиданням на диск. За увімкненої буферизації запит може бути переданий на бекенд лише після повного отримання, а при буферизації відповідей бекенд звільняє потік виконання одразу після формування відповіді. Це підвищує пропускну здатність і захищає бекенд, але водночас збільшує затримки, споживання пам’яті та кількість операцій введення-виведення (що, втім, менш критично при використанні SSD);
- повне вимкнення буферизації, що доцільно для маршрутів, чутливих до затримок, зокрема у сценаріях потокового передавання даних. Однак у цьому випадку бекенд повинен коректно працювати з повільними клієнтами, включно із захистом від атак типу Slow POST або Slow Read;
- керована буферизація на рівні застосунку, яку можна реалізувати за допомогою заголовка X-Accel-Buffering, дозволяючи прикладному коду визначати, чи слід буферизувати конкретну відповідь.

Незалежно від обраної стратегії, усі зміни необхідно обов’язково перевіряти з точки зору TTFB і TTLB. Буферизація може істотно впливати не лише на затримки, але й на кількість операцій введення-виведення та рівень навантаження на бекенд, тому ці показники також слід постійно моніторити під час експериментів і впровадження оптимізацій.

3 ВИКОРИСТАННЯ ІНФРАСТРУКТУРНОГО КОМПОНЕНТУ KAFAKA CONNECT

Kafka Connect використовується як інфраструктурний компонент потокової обробки даних, який забезпечує надійну, масштабовану та відмовостійку доставку фрагментів відеопотоку, а також збереження супровідних метаданих, таких як часові мітки, ідентифікатори кадрів і інформація про втрати пакетів. Завдяки механізмам повторної доставки, збереження порядку повідомлень і асинхронної взаємодії між компонентами системи, Kafka Connect створює стабільне середовище для подальшої обробки відеоданих навіть за умов нестабільних мереж із обмеженою пропускною здатністю.

Алгоритмічне відновлення відео після втрат реалізується на вищому рівні обробки за допомогою моделей деконволюції, які використовують отримані фрагменти та метадані для реконструкції просторово-часової структури відеопотоку. Таким чином, Kafka Connect виконує роль надійного транспортно-буферного шару, що забезпечує коректну роботу деконволюційних алгоритмів у розподілених системах обробки відео.

3.1 Основні налаштування

Коли йдеться про налаштування Source-конекторів, насамперед необхідно чітко розуміти принцип їхньої роботи та внутрішню архітектуру Kafka Connect. Для цього доцільно розглянути типовий приклад JDBC Source-конектора, логічну структуру якого можна умовно поділити на кілька взаємопов’язаних етапів.

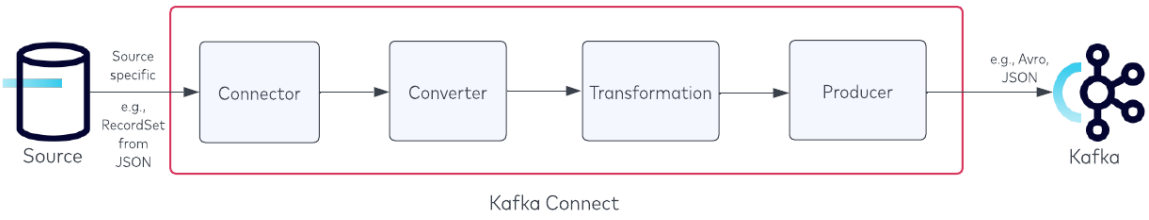


Рис. 3.1 Архітектура Source-конектора в Kafka Connect

Ілюструє послідовність обробки даних від джерела (Source) через конектор, конвертацію форматів, застосування трансформацій і подальшу передачу повідомлень продюсером до топіків Kafka

На першому етапі існує кінцева точка або джерело даних, з якого здійснюється зчитування записів. У розглянутому прикладі таким джерелом виступає база даних. Далі в процес вступає Kafka Connect, усередині якого виконуються такі функціональні компоненти:

- **Конектор** — відповідає за підключення до джерела даних і повертає певну кількість записів, отриманих із цього джерела. Саме на цьому рівні визначається логіка опитування, інкрементального читання та обробки змін у даних.
- **Конвертер** — перетворює отримані записи у вибраний формат представлення даних, наприклад Avro, Protobuf або JSON. Цей етап є обов’язковим, оскільки забезпечує уніфікований формат повідомлень для подальшої передачі та обробки.
- **Перетворення (трансформації)** — застосовує прості перетворення повідомлень (Simple Message Transformations, SMT), які визначаються в конфігурації конектора. На цьому етапі можуть виконуватися операції фільтрації, перейменування полів, додавання або видалення атрибутів.
- **Producer** — відповідає за відправлення підготовлених записів у відповідний топік Kafka з урахуванням налаштувань продуктивності та надійності.

Після цього керування переходить до Kafka, яка приймає повідомлення від продюсера, зберігає їх у відповідних топіках і забезпечує подальше споживання іншими компонентами системи.

Після розгляду базової архітектури Kafka Connect важливо окремо зазначити, які аспекти не підлягають налаштуванню з точки зору продуктивності та затримок:

- **Конвертер** — перетворення записів у формати Avro, Protobuf або JSON завжди потребує певного часу виконання, і цього накладного часу неможливо повністю уникнути.

- **Трансформації** — хоча SMT виконуються дуже швидко, їх застосування все одно додає додаткову затримку. Якщо необхідно мінімізувати латентність, можна відмовитися від використання трансформацій або звести їх кількість до мінімуму.

Водночас існує низка компонентів, які можна і доцільно налаштовувати для оптимізації роботи Source-конекторів:

- **Конектор** — за допомогою його конфігурацій можна керувати частотою опитування джерела, розміром вибірки, режимами інкрементального читання та іншими параметрами.
- **Producer** — параметри продюсера, зокрема batch.size, linger.ms та інші, можуть бути використані для підвищення пропускної здатності та зменшення накладних витрат на передачу повідомлень.
- **Джерело даних** — у цьому контексті не розглядається, оскільки його продуктивність і поведінка повністю залежать від конкретного типу джерела (база даних, файлове сховище, API тощо).
- **Сама Kafka** — також не є предметом детального аналізу в межах цього опису, однак її конфігурація безпосередньо впливає на загальну ефективність системи; для поглибленого вивчення цього питання доцільно звернутися до спеціалізованих керівництв і документації.

Таким чином, ефективне налаштування Source-конекторів у Kafka Connect полягає не в усуненні неминучих накладних витрат, а в грамотному балансуванні параметрів конектора та продюсера з урахуванням характеристик джерела даних і вимог до пропускної здатності та затримок у конкретній системі.

Налаштування конекторів

Налаштування конектора безпосередньо визначається тим, які конфігураційні параметри він надає. Якщо конкретний конектор не містить параметрів, що дозволяють збільшити обсяг даних, які зчитуються з джерела за один цикл, то можливості підвищення пропускної здатності на цьому рівні будуть суттєво обмежені. Інакше кажучи, конектор не можна «прискорити» понад ті межі, які закладені його реалізацією та відкритими параметрами конфігурації.

Для прикладу розглянемо Confluent JDBC Source-конектор, який надає низку важливих параметрів, зокрема:

- batch.max.rows — визначає максимальну кількість рядків, які можуть бути включені в один пакет під час зчитування нових даних із джерела;
- poll.interval.ms — задає інтервал у мілісекундах, з яким конектор опитує таблиці бази даних на предмет появи нових або змінених записів.

Збільшуючи значення batch.max.rows, можна змусити конектор повертати продюсеру більше записів за один запит, що безпосередньо впливає на загальну пропускну здатність системи. Якщо ж конектор не надає такого параметра і має, наприклад, фіксоване значення batch.max.rows = 5, то незалежно від навантаження максимальна кількість записів, отриманих за один запит, завжди обмежуватиметься цими п’ятьма записами. Таким чином, конектор може бути налаштований лише в межах доступних йому параметрів, і будь-які оптимізації повинні відштовхуватися саме від них.

Конфігураційні параметри Producer, які можна змінювати

Під час налаштування Kafka Producer зазвичай розглядають кілька ключових параметрів, які безпосередньо впливають на продуктивність, затримки та стабільність роботи:

- **batch.size** — задає максимальний розмір пакета повідомлень у байтах (типове значення — 16384). Чим більший пакет, тим ефективніше використовується мережа, але тим вищі затримки;
- **linger.ms** — визначає максимальний час у мілісекундах, протягом якого продюсер очікує на наповнення пакета перед його відправленням (типове значення — 0);
- **buffer.memory** — загальний обсяг пам’яті в байтах, який продюсер може використовувати для буферизації повідомлень, що очікують відправлення на брокер Kafka (типове значення — 33554432, тобто 32 МБ);
- **compression.type** — задає тип стиснення повідомлень для топіка (за замовчуванням — "producer").

Параметр linger.ms тісно пов’язаний із тим, яку кількість записів конектор передає продюсеру. Якщо за один цикл надходить, наприклад, лише 10 записів, продюсеру може знадобитися зовсім невеликий час очікування — умовно 5 мс. Водночас у сценарії, коли конектор повертає десятки або сотні тисяч записів, значення linger.ms доцільно збільшити, оскільки пакету потрібно більше часу, щоб заповнитися до оптимального розміру. Якщо linger.ms встановлено занадто малим, пакети часто відправлятимуться напівпорожніми, що знижує ефективність. Якщо ж це значення надмірно велике, продюсер чекатиме без реальної необхідності, і загальна швидкість доставки повідомлень зменшиться.

Параметр buffer.memory визначає, скільки пам’яті продюсер може використати для тимчасового зберігання пакетів, які ще не були відправлені брокеру. Це значення має приблизно відповідати очікуваному обсягу буферизації, однак воно не є жорсткою межею для всієї пам’яті, яку використовує продюсер. У ситуації, коли Kafka-брокер недоступний або не встигає приймати повідомлення, продюсер починає накопичувати пакети в буфері. Після заповнення буфера він очікує протягом max.block.ms (типово 60 000 мс), доки буфер не звільниться. Якщо цього не відбувається, генерується виняток. Надто мале значення buffer.memory призводить до швидкого переповнення буфера і частих помилок, тоді як занадто велике значення може спричинити вичерпання оперативної пам’яті системи та помилки типу OOM.

Останній важливий параметр — compression.type, який відповідає за стиснення повідомлень перед їх відправленням. Кожен алгоритм стиснення має свої переваги та недоліки з точки зору швидкості, ступеня стиснення і навантаження на CPU. Тому вибір оптимального типу стиснення потребує окремого експериментального аналізу для конкретного сценарію використання. Хоча стиснення дозволяє значно зменшити обсяг передаваних даних, воно неминуче збільшує затримку доставки, оскільки повідомлення необхідно додатково стискати перед відправленням і розпаковувати на стороні споживача.

У підсумку, ефективне налаштування Kafka Connect і Producer полягає у збалансованому підборі параметрів, який враховує можливості конектора, характер

джерела даних, вимоги до пропускної здатності та допустимі затримки в конкретній системі.

3.2 JMX метрики

JMX-метрики слід розділити на три секції: метрики конектора, метрики брокера та метрики продюсера (з Kafka Connect фреймворку).

Таблиця 3.1

JMX-метрики

Рівень	**Метрика**	**Опис**
Коннектор	source-record-poll-rate	Середня кількість записів за секунду, які задача (task) Source-конектора зчитує з джерела або генерує до застосування трансформацій у межах робочого процесу (worker).
Коннектор	poll-batch-avg-time-ms	Середній час у мілісекундах, який задача витрачає на отримання одного пакета записів від джерела.
Коннектор	source-record-write-rate	Середня кількість записів за секунду, які після застосування трансформацій записуються в Kafka (без урахування записів, відфільтрованих SMT).
Брокер	kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec	Швидкість надходження байтів від клієнтів до брокера Kafka для конкретного топіка.
Продюсер	record-size-avg	Середній розмір одного запису, який відправляється продюсером.
Продюсер	batch-size-avg	Середня кількість байтів, що відправляються у партицію за один запит продюсера.
Продюсер	records-per-request-avg	Середня кількість записів, які надсилаються в одному запиті продюсера.
Продюсер	record-send-rate	Середня кількість записів, що надсилаються продюсером у топік за секунду.

Приклад покрокового налаштування

У цьому прикладі демонструється підхід до поетапного підвищення пропускної здатності Confluent JDBC Source-конектора, який підключається до бази даних MySQL. Описана методологія є універсальною і може бути застосована практично до будь-якого Source-конектора, незалежно від типу джерела даних. Основна ідея полягає в тому, щоб послідовно визначити вузькі місця, зафіксувати початкові характеристики системи та лише після цього вносити зміни до конфігурацій.

Для досягнення цієї мети доцільно виконати такі кроки:

- визначити, які конфігураційні параметри конектора та продюсера можна змінювати, а також налаштувати Grafana для візуалізації JMX-метрик;
- зафіксувати початкову пропускну здатність системи до внесення будь-яких змін;
- визначити, на якому рівні доцільніше виконувати оптимізацію — на рівні продюсера чи на рівні конектора;
- змінити конфігурацію продюсера з метою підвищення ефективності пакетування та передачі повідомлень;
- змінити конфігурацію конектора для збільшення обсягу даних, які зчитуються з джерела за одиницю часу.

Крок 1. Визначення доступних конфігурацій і налаштування моніторингу

На першому етапі необхідно чітко окреслити перелік параметрів, які можуть бути змінені без модифікації коду, а також підготувати інструменти спостереження за роботою системи. Для цього використовується збір JMX-метрик і їх відображення в Grafana, що дозволяє в реальному часі оцінювати вплив кожної зміни на продуктивність.

Для JDBC Source-конектора ключовими конфігураціями є:

- **batch.max.rows** — максимальна кількість рядків, які можуть бути включені в один пакетний запит конектора до бази даних. Цей параметр безпосередньо

впливає на обсяг даних, що буферизуються та передаються за один цикл опитування (типове значення — 100);

- **poll.interval.ms** — інтервал у мілісекундах, протягом якого конектор очікує перед наступним запитом нових даних для кожної таблиці (типове значення — 5000).

З боку **Kafka Producer** до параметрів, що найчастіше підлягають оптимізації, належать:

- **batch.size** — максимальний розмір пакета повідомлень у байтах (типове значення — 16384);
- **linger.ms** — максимальний час очікування наповнення пакета перед його відправленням (типове значення — 0);
- **buffer.memory** — загальний обсяг пам’яті, який продюсер може використати для буферизації повідомлень, що очікують на відправлення (типове значення — 33554432);
- **compression.type** — тип стиснення, який застосовується до повідомлень перед їх передаванням у Kafka (за замовчуванням не заданий).

У межах цього прикладу використовується середовище kafka-docker-playground, яке значно спрощує відтворення експерименту. Спеціальний скрипт mysql.sh розгортає екземпляр MySQL, заповнює базу даних 10 мільйонами записів і запускає JDBC Source-конектор разом із Grafana. Для цього в середовищі додатково активується експорт JMX-метрик шляхом встановлення параметра ENABLE_JMX_GRAFANA=true.

Такий підхід дозволяє не лише спостерігати за поточною пропускною здатністю, але й наочно оцінювати, як зміна кожного окремого параметра впливає на загальну продуктивність конектора та всієї потокової системи.

Крок 2 — Визначення початкової пропускної здатності

На другому етапі необхідно зафіксувати вихідну (базову) пропускну здатність системи при використанні стандартних, тобто початкових, конфігурацій конектора та продюсера. Це є критично важливим кроком, оскільки без чіткого розуміння

початкових показників неможливо об’єктивно оцінити ефективність подальших оптимізацій.

Для цього слід запустити конектор із типовими налаштуваннями та зібрати відповідні JMX-метрики, які відображають швидкість зчитування даних із джерела, час опитування, обсяг переданих записів і загальну інтенсивність трафіку. Отримані значення слугуватимуть точкою відліку, з якою можна буде порівнювати показники після внесення змін до конфігурації. Саме порівняння метрик «до» і «після» дозволяє точно визначити, які налаштування дійсно призводять до зростання пропускної здатності, а які не дають відчутного ефекту.

Важливо враховувати, що на етапі налаштування доцільно працювати лише з однією задачею (task) конектора. Це спрощує аналіз і дозволяє ізолювати вплив кожного параметра, не змішуючи його з ефектами паралельного виконання. Після того як базова пропускна здатність для однієї задачі буде чітко визначена й оптимізована, кількість задач можна збільшити пізніше шляхом налаштування параметра task.max — за умови, що конкретний конектор підтримує масштабування на рівні задач.

Такий поетапний підхід забезпечує контрольовану оптимізацію, зменшує ризик хибних висновків і дозволяє будувати подальші рішення на основі достовірних вимірювань, а не припущень.

Базові JMX-метрики. Брокер

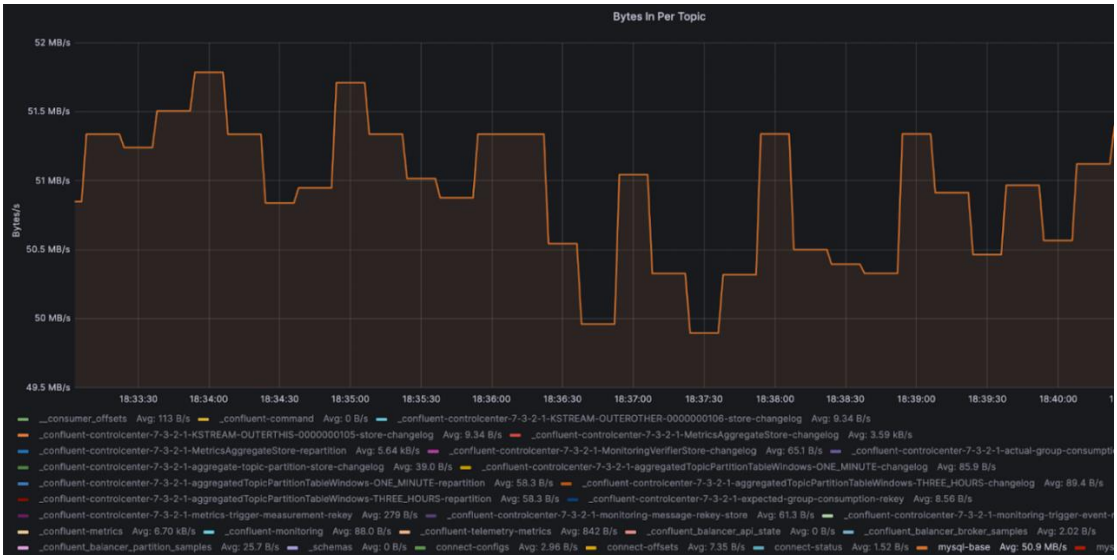


Рис. 3.2 BytesInPerSec - 50.9 MB/s

Конектор

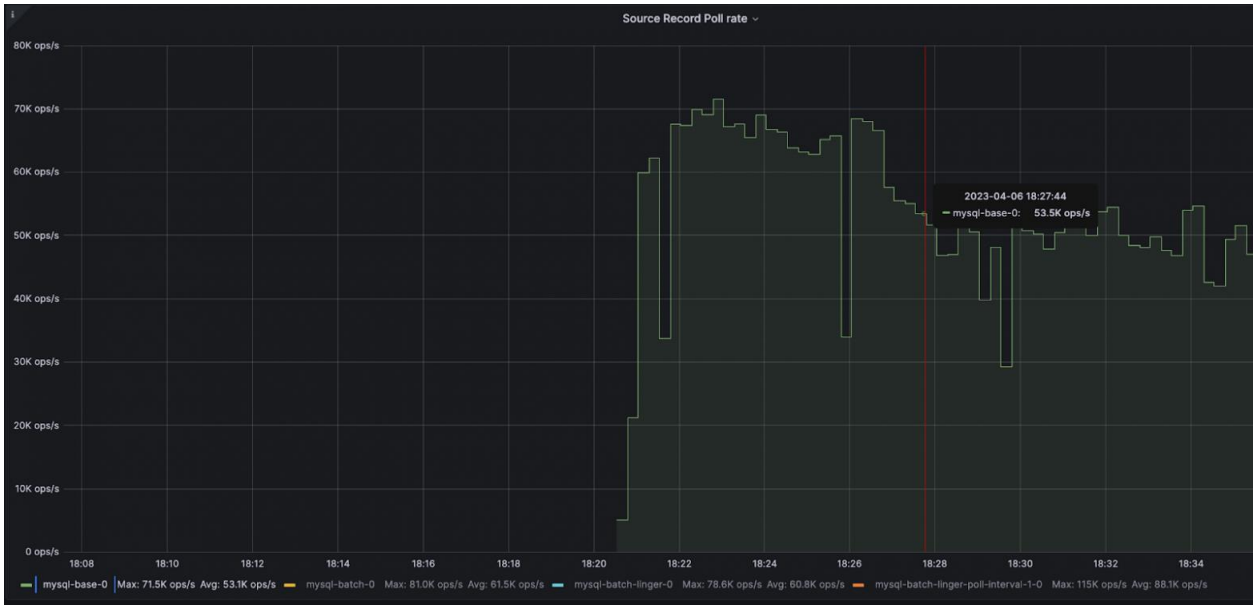


Рис. 3.3 source-record-poll-rate - 53.1K в секунду

Продюсер



Рис. 3.4 record-size-avg - 1.09 KiB

Крок 3. Визначення конфігурацій, які доцільно змінювати (продюсер чи конектор)

На цьому етапі необхідно з’ясувати, на якому рівні оптимізація буде найбільш ефективною — на рівні продюсера або на рівні конектора. Для цього слід проаналізувати зібрані раніше метрики та узагальнити їх у зручному для аналізу вигляді. Найпростіший і наочний спосіб зробити це — звести основні показники в

узагальнювальну таблицю, яка дозволить швидко визначити вузькі місця та прийняти обґрунтоване рішення щодо подальших змін конфігурації.

Таблиця 3.2

Порівняння метрик

Метрика	Значення
BytesInPerSec	50.9 MB/s
source-record-poll-rate	53.1K ops/s
record-size-avg	1.09 KiB

На основі наведених вище метрик першочергову увагу доцільно зосередити саме на метриках рівня Kafka Producer. Причина такого вибору полягає в аналізі JMX-показників batch-size-avg та records-per-request-avg, які чітко вказують на наявність обмежень на етапі формування та відправлення пакетів.

Зокрема, значення batch-size-avg становить приблизно 15,1 KiB, що практично дорівнює типовому граничному значенню параметра batch.size, який за замовчуванням дорівнює 16384 байтам. За таких умов додавання ще одного запису до пакета стає неможливим, оскільки середній розмір одного запису складає близько 1,09 KiB. Таким чином, пакет досягає максимально допустимого розміру ще до того, як буде повністю використано потенціал агрегації повідомлень.

Аналогічну картину демонструє метрика records-per-request-avg, яка показує, що в середньому за один запит продюсер відправляє лише близько 14,8 записів. Водночас конфігурація конектора дозволяє повертати до 100 записів за один цикл опитування завдяки параметру batch.max.rows. Це означає, що конектор фактично зчитує та передає значно більший обсяг даних, однак продюсер не здатний ефективно агрегувати ці записи в одному запиті через жорстке обмеження на розмір пакета.

Отже, спостерігається ситуація, коли продуктивність конектора перевищує можливості продюсера, а параметр batch.size стає основним стримувальним фактором. Це чітко вказує на доцільність початку оптимізації саме з налаштування продюсера, оскільки без збільшення розміру пакета або зміни параметрів його

формування подальше підвищення пропускної здатності на рівні конектора не приведе до очікуваного ефекту.

Спираючись на результати аналізу метрик рівня продюсера, логічним першим кроком оптимізації є налаштування саме продюсера. Саме на цьому рівні найчастіше виникають обмеження, пов’язані з формуванням пакетів, буферизацією та ефективністю відправлення повідомлень до брокера Kafka.

Водночас після внесення змін до конфігурації продюсера постає закономірне запитання: як визначити, що конектор не є вузьким місцем у системі? На практиці існує кілька характерних ознак, які дозволяють зробити такий висновок. Зазвичай спостерігається один або одразу обидва з наведених нижче сценаріїв:

- загальна пропускна здатність системи не зростає навіть після зміни параметрів продюсера;
- при зміні конфігурацій продюсера значення метрики send-rate залишається незмінним або не демонструє очікуваного збільшення.

Поява одного або обох цих сценаріїв є чітким сигналом того, що подальша оптимізація продюсера не дає ефекту, і увагу необхідно перенести на конектор. Метрика send-rate, що надається через JMX, відображає фактичну активність продюсера в поточний момент часу. Якщо її значення не зростає або змінюється незначно, це означає, що продюсер не обмежений власними ресурсами чи налаштуваннями, а очікує надходження нових записів від конектора.

У такій ситуації продюсер фактично переходить у режим очікування, а вузьким місцем стає саме конектор, який не встигає постачати достатню кількість даних. Отже, стабільно низьке або незмінне значення send-rate після оптимізації продюсера є надійним індикатором того, що необхідно модифікувати конфігурацію або логіку роботи конектора з метою підвищення загальної пропускної здатності потокової системи.

Крок 4. Зміна конфігурацій продюсера

На цьому етапі виконується безпосереднє коригування параметрів Kafka Producer з метою подальшого підвищення пропускної здатності системи. У поточній конфігурації параметр batch.size фактично вже повністю заповнюється

записами, які надходять від конектора, що обмежує можливість додаткової агрегації повідомлень у межах одного запиту. Тому в даному прикладі доцільно збільшити значення batch.size, а разом із ним — і параметр batch.max.rows на стороні конектора.

Причина одночасного налаштування обох параметрів полягає не лише у прагненні збільшити пропускну здатність, а й у практичному бажанні уникнути повторного перерахунку batch.size на наступних етапах оптимізації. Оскільки наперед відомо, що необхідно отримувати більший обсяг даних із джерела за один цикл опитування, логічно одразу збільшити як кількість записів, що повертаються конектором, так і максимально допустимий розмір пакета на рівні продюсера.

Таке налаштування також дозволяє усунути конектор як потенційне вузьке місце системи, оскільки тепер він буде запитувати та передавати продюсеру значно більшу кількість повідомлень. У межах цього прикладу для параметра batch.max.rows встановлюється значення 500 записів, що суттєво перевищує початкову конфігурацію і створює умови для більш ефективного використання механізмів пакетування.

Для коректного вибору нового значення batch.size необхідно орієнтуватися на середній розмір пакета. При цьому слід враховувати, що метрика batch-size-avg у JMX зазвичай відображається в одиницях KiB. Для переходу до байтів це значення потрібно помножити на 1024, що дозволяє точно оцінити фактичний обсяг даних і задати параметр batch.size таким чином, щоб він забезпечував максимальне заповнення пакета без перевищення допустимих меж. Такий підхід створює основу для подальшого стабільного зростання пропускної здатності всієї потокової системи.

На цьому етапі конфігурація конектора буде оновлена, і після внесення змін має бути помітно, що значення параметрів batch.max.rows та batch.size були збільшені. Необхідність такого коригування знову ж таки обґрунтовується аналізом JMX-метрики batch-size-avg, яка показує, що наявний пакет уже повністю заповнюється 100 записами. Саме це підтверджує доцільність розширення

обмежень на розмір пакета з метою подальшого підвищення пропускної здатності та більш ефективного використання можливостей продюсера і конектора.

```
curl -X PUT \
     -H "Content-Type: application/json" \
     --data '{
               "connector.class":"io.confluent.connect.jdbc.JdbcSourceConnector",
               "tasks.max":"1",
               "connection.url":"jdbc:mysql://mysql:3306/mydb?user=user&password=password&use
               "mode":"bulk",
               "batch.max.rows": 500,
               "producer.override.client.id": "mysql-batch",
               "producer.override.batch.size": 558080,
               "query":"SELECT * FROM mydb.team WHERE mydb.team.id < 7900000 ",
               "topic.prefix":"mysql-batch"
          }' \
```

Рис. 3.5 Параметри batch.max.rows та batch.size збільшилися

Після запуску конектора ви отримаєте наступний зведений результат JMX:

Таблиця 3.3

Оновлені дані метрик

Метрика	Значення
BytesInPerSec	50.9 MB/s
source-record-poll-rate	53.1K ops/s
record-size-avg	1.09 KiB

Крок 5. Зміна конфігурації конектора для підвищення пропускної здатності

На попередньому етапі було виконано налаштування продюсера та встановлено, що поточним вузьким місцем системи є саме конектор. Такий висновок зроблено на основі того, що після оптимізації продюсера загальна пропускна здатність не зросла, а значення метрики record-send-rate залишалося відносно стабільним. Це свідчить про те, що продюсер більше не обмежує швидкість передавання даних і змушений очікувати надходження нових записів від конектора.

З боку конектора існує обмежена кількість параметрів, які безпосередньо впливають на його продуктивність і можуть бути використані для подальшого збільшення пропускної здатності. До таких параметрів належать:

- batch.max.rows — максимальна кількість рядків, які можуть бути включені в один пакет під час запиту нових даних. Цей параметр визначає обсяг даних, що буферизуються та повертаються конектором за один цикл опитування;
- poll.interval.ms — інтервал у мілісекундах між послідовними запитами нових даних для кожної таблиці, тобто частота, з якою конектор звертається до джерела.

Оскільки значення batch.max.rows уже було збільшено на попередньому кроці і при цьому не спостерігалося зростання пропускної здатності, основним кандидатом для подальшої оптимізації залишається параметр poll.interval.ms. Саме він визначає, як часто конектор виконує запити до бази даних. Поточне значення цього параметра дорівнює 5000 мс, що означає, що конектор отримує нові дані з таблиці лише один раз на п’ять секунд.

За таких умов навіть збільшений розмір пакетів не може повністю реалізувати свій потенціал, оскільки конектор фізично не встигає постачати дані з достатньою частотою. Отже, зменшення poll.interval.ms є логічним наступним кроком, який дозволяє підвищити інтенсивність зчитування даних із джерела та, як наслідок, збільшити загальну пропускну здатність потокової системи.

4 ПІДВИЩЕННЯ ПРОПУСКНОЇ ЗДАТНОСТІ ДЛЯ УСУНЕННЯ ВТРАТ В МЕРЕЖІ

Стандарт DOCSIS 3.1 визначає фізичні та канальні механізми передавання даних у кабельних мережах, які характеризуються обмеженою та динамічно змінною пропускною здатністю. Особливості роботи таких мереж, зокрема вплив шумів, перевантажень і спільного використання середовища передавання, можуть призводити до втрат пакетів, коливань затримок і нестабільності швидкості передавання даних. Зазначені обмеження на нижчих рівнях моделі мережевої взаємодії створюють передумови для застосування додаткових методів обробки даних на прикладному рівні. Зокрема, використання алгоритмів деконволюції дозволяє відновлювати втрачені або спотворені фрагменти відеопотоків, компенсувати негативний вплив мережевих обмежень і, як наслідок, підвищувати якість відеосервісів та загальний рівень сприйняття якості обслуговування користувачами.

4.1 Специфікація

Нова версія специфікації DOCSIS — DOCSIS 3.1 суттєво змінила принципи функціонування кабельних мереж доступу, зробивши значний крок уперед у порівнянні з попередніми поколіннями стандарту. Зокрема, впровадження DOCSIS 3.1 дозволило збільшити ефективну пропускну здатність каналу приблизно на 50 %, забезпечивши теоретичну продуктивність до 10 Гб/с у прямому каналі та до 2 Гб/с у зворотному каналі. Такі показники вже є порівнюваними зі швидкостями передавання даних в оптоволоконних мережах, що значно розширює можливості кабельної інфраструктури для надання високоякісних мультимедійних сервісів.

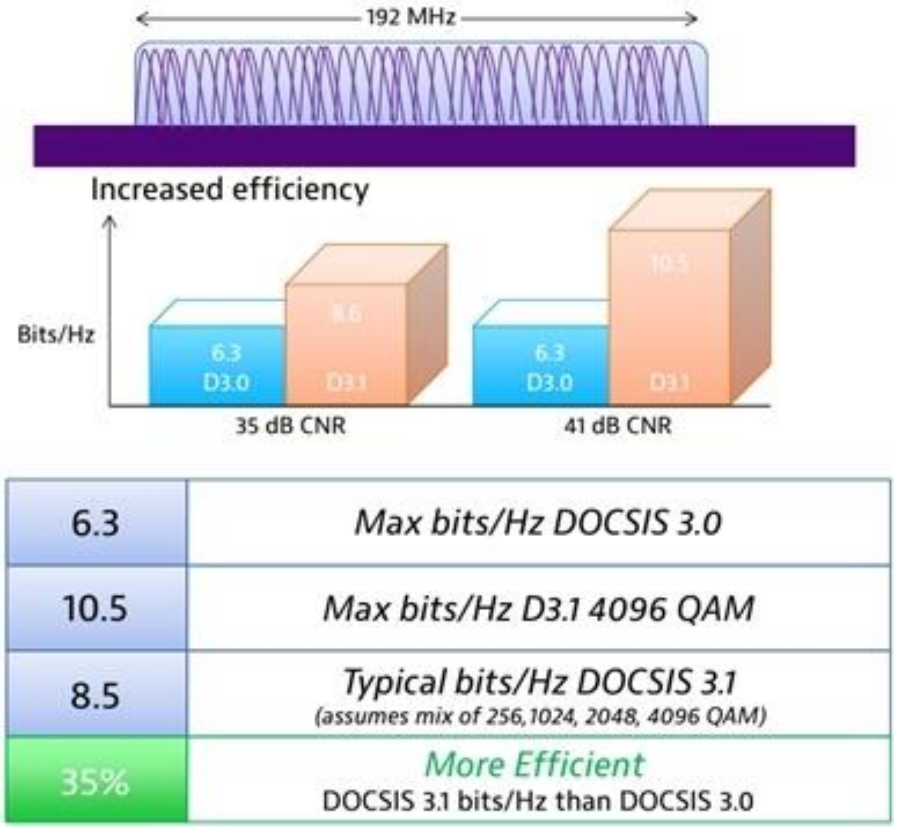


Рис. 4.1 Порівняння спектральної ефективності стандартів DOCSIS 3.0 та DOCSIS 3.1, що ілюструє збільшення кількості переданих бітів на 1 Гц смуги частот завдяки використанню OFDM і високорівневої модуляції (до 4096-QAM) у DOCSIS 3.1

Однією з ключових переваг DOCSIS 3.1 є підвищення спектральної ефективності: стандарт забезпечує більшу кількість бітів, переданих на 1 Гц смуги частот, у порівнянні з DOCSIS 3.0 за однакового співвідношення сигнал/шум. Це означає більш раціональне використання доступного частотного ресурсу, що особливо важливо для мереж із високою щільністю абонентів та змінними умовами передавання.

Специфікація DOCSIS 3.1 була офіційно представлена та успішно протестована в лабораторних умовах у 2015 році. Уже на початку 2016 року було сертифіковано перші кабельні модеми, що підтримують цей стандарт, а провайдери в різних країнах світу розпочали активне впровадження та польові випробування відповідного обладнання. Це стало важливим етапом переходу кабельних мереж до нових рівнів швидкості та якості обслуговування.

Унікальність DOCSIS 3.1 у порівнянні з попередніми версіями полягає не лише в підвищенні швидкостей, а й у використанні принципово нових технологічних підходів. У межах специфікації ключову роль відіграють дві базові технології: ортогональне частотне мультиплексування (OFDM) та коди з малою щільністю перевірок на парність (LDPC). OFDM забезпечує гнучкий поділ частотного спектра на велику кількість вузьких піднесучих, що підвищує стійкість до завад і селективних спотворень каналу. LDPC, у свою чергу, реалізує потужні механізми корекції помилок, значно зменшуючи ймовірність втрат даних навіть у складних умовах передавання.

Разом із тим, незважаючи на вдосконалені механізми фізичного та канального рівнів, у реальних мережах DOCSIS 3.1 все ще можливі втрати пакетів, коливання затримок і деградація якості відеопотоків, особливо за пікових навантажень або несприятливих умов каналу. Саме в таких сценаріях важливу роль відіграють алгоритми деконволюції для відновлення відеопотоків, які застосовуються вже на прикладному рівні. Підвищена пропускна здатність і зменшена кількість помилок, що забезпечуються DOCSIS 3.1, створюють більш сприятливі вхідні умови для роботи деконволюційних моделей, дозволяючи їм ефективніше реконструювати втрачені або спотворені фрагменти відео та підвищувати суб’єктивну якість відтворення.

Таким чином, DOCSIS 3.1 не лише розширює технічні можливості кабельних мереж, а й опосередковано підвищує ефективність сучасних методів обробки мультимедійних даних, зокрема алгоритмів деконволюції, які використовуються для відновлення відеопотоків у мережах з обмеженою та динамічною пропускною здатністю.

Ортогональне частотне мультиплексування (OFDM)

Найпростіше зрозуміти принцип роботи OFDM, порівнявши його з підходом, який використовувався у стандарті DOCSIS 3.0. У DOCSIS 3.0 для одного прямого каналу застосовується єдина несуча частота з шириною смуги 6 МГц (або 8 МГц у європейських мережах). Для модуляції цієї несучої використовується одноканальна квадратурна амплітудна модуляція SC-QAM, а передавання символів відбувається

строго послідовно на одній частоті. У разі погіршення умов прийому сигналу оператор змушений зменшувати рівень модуляції, причому не лише для проблемної ділянки, а фактично для всіх каналів мережі. Таким чином, параметри модуляції оптимізуються під найгіршу ділянку коаксіальної мережі, що істотно обмежує загальну пропускну здатність.

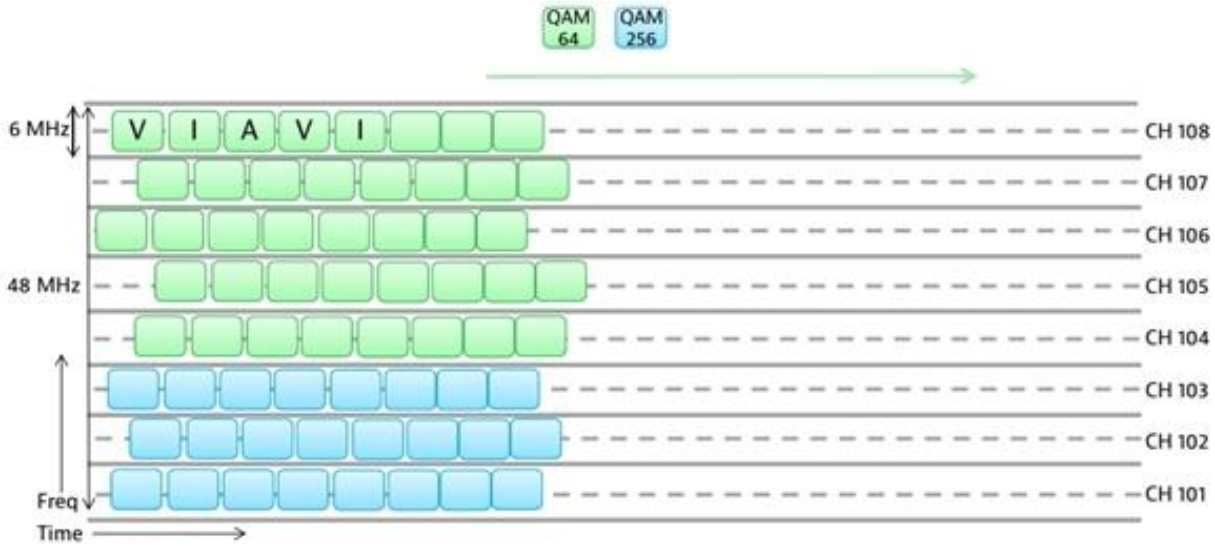


Рис. 4.2 Порівняння традиційної модуляції SC-QAM та ортогонального частотного мультиплексування OFDM, що ілюструє розподіл символів у часі та частоті, а також гнучке використання піднесучих і різних рівнів QAM у DOCSIS 3.1

На відміну від цього підходу, OFDM використовує значно ширшу смугу частот — від 24 до 192 МГц. Усередині цієї смуги формується велика кількість вузьких піднесучих — до кількох тисяч, кожна з яких має ширину 25 або 50 кГц. Усі піднесучі синхронізовані між собою в часі та працюють як єдина система. Символи розподіляються між піднесучими та часовими інтервалами, утворюючи кодові слова, які передаються паралельно по всьому частотному діапазону.

Ключова перевага такого підходу полягає в тому, що дані передаються одночасно на багатьох частотах. Це відкриває нові можливості адаптації до умов каналу. Якщо на окремій піднесучій виникають завади або зростає рівень шуму, система може знизити рівень модуляції лише для цієї піднесучої або взагалі виключити її з використання, компенсуючи втрату за рахунок сусідніх частот. Передавання при цьому не зупиняється і продовжується з оптимальним рівнем продуктивності. Такий механізм робить OFDM значно менш чутливим до

вузькосмугових і імпульсних завад, оскільки вони впливають лише на окремі піднесучі, а не на весь спектр сигналу.

Додатковою перевагою OFDM є можливість точного керування фазами піднесучих. Завдяки цьому зменшується взаємна інтерференція між сусідніми піднесучими, що дозволяє використовувати вищі рівні модуляції і, відповідно, підвищувати загальну пропускну здатність мережі. На відміну від SC-QAM, де для всього каналу застосовується один рівень модуляції, OFDM дозволяє призначати індивідуальний рівень модуляції для кожної піднесучої. Для цього використовуються спеціальні модуляційні профілі, які можуть змінюватися залежно від стану каналу.

Кожна піднесуча може працювати з власним рівнем модуляції — наприклад, 64 QAM, 1024 QAM, 2048 QAM або 4096 QAM. У межах одного сегмента HFC-мережі певна піднесуча може використовувати максимальний рівень 4096 QAM, тоді як в іншому сегменті — лише 1024 QAM, а в умовах сильної інтерференції окремі частини спектра можуть бути повністю виключені з профілю. Таким чином, система динамічно адаптується до реальних умов передавання даних.

Усі піднесучі працюють спільно, формуючи кодові слова, а модуляційні профілі (умовно позначені, наприклад, літерами A, B, C, D) визначають параметри модуляції для кожної піднесучої. Завдяки цьому оптимізація виконується не лише для окремих частот, а для всього набору піднесучих у комплексі, що істотно підвищує ефективність використання спектра.

Замість оптимізації під найгіршу ділянку мережі, як це було в попередніх стандартах, DOCSIS 3.1 дозволяє оптимізувати передавання під найкращі доступні умови в кожний момент часу. Саме це робить дану технологію значно ефективнішою. Якщо в DOCSIS 3.0 типовий канал забезпечував близько 6,3 біта на 1 Гц, то DOCSIS 3.1 здатен досягати до 10,5 біт на 1 Гц при використанні модуляції 4096 QAM. У більш реалістичних умовах, коли одночасно застосовуються різні рівні модуляції, досягається близько 8,5 біт на 1 Гц, що відповідає підвищенню спектральної ефективності приблизно на 35 % без змін у фізичній HFC-інфраструктурі.

Підвищення пропускної здатності та стабільності каналу безпосередньо впливає і на якість передавання відеопотоків. Менша кількість втрат пакетів і більш рівномірне передавання даних знижують імовірність появи артефактів і пропусків кадрів. Це, у свою чергу, зменшує навантаження на алгоритми відновлення відеопотоків, зокрема алгоритми деконволюції, які застосовуються на прикладному рівні для реконструкції втрачених або спотворених фрагментів зображення. Таким чином, зростання пропускної здатності, забезпечене OFDM у DOCSIS 3.1, не лише підвищує ефективність мережі, а й опосередковано сприяє покращенню якості відновленого відео в умовах обмежених або нестабільних мереж.

Коди з малою щільністю перевірок на парність (LDPC)

Покращення, досягнуті завдяки використанню OFDM, були б неможливими без застосування сучасних алгоритмів корекції помилок. У стандарті DOCSIS 3.0 для упреждаючої корекції помилок використовувався код Ріда–Соломона (FEC), а якість каналу оцінювалася за показником BER (Bit Error Rate). Такий підхід був логічним для систем із однією несучою частотою, однак у випадку OFDM, де передавання даних відбувається одночасно на великій кількості піднесучих, оцінка помилок лише на рівні окремих бітів або однієї несучої втрачає свою інформативність.

У стандарті DOCSIS 3.1 замість класичного FEC застосовується алгоритм LDPC (Low Density Parity Check). Його принципова відмінність полягає в тому, що корекція помилок виконується по всьому частотному діапазону, а аналіз здійснюється не для окремих бітів, а для кодового слова в цілому. Якщо помилка в кодовому слові підлягає виправленню, алгоритм LDPC автоматично її коригує без необхідності повторної передачі даних. Це суттєво зменшує кількість повторних передавань і дозволяє використовувати вищі рівні модуляції, навіть у складних умовах каналу.

Завдяки цьому LDPC дає змогу максимально ефективно використовувати доступну пропускну здатність і наближати реальні характеристики каналу до теоретичних меж, визначених теоремою Шеннона. У практичному вимірі це означає вищу швидкість передавання, меншу ймовірність втрат пакетів і

стабільнішу роботу мережі, що особливо важливо для передавання потокового відео.

Разом із тим алгоритм LDPC має і певні обмеження. Оскільки він працює в режимі динамічної адаптації, система може в реальному часі підлаштовувати рівні модуляції та потужності сигналу, компенсуючи зростання кількості помилок. У результаті мережа може тривалий час функціонувати на граничних режимах без явних ознак деградації з боку оператора. Проте в певний момент рівень завад може перевищити можливості корекції, і помилки стають некоректованими, що вже безпосередньо відчувається кінцевими користувачами у вигляді зниження якості сервісу.

У контексті передавання відеопотоків така ситуація може призводити до втрати фрагментів кадрів, появи артефактів або збоїв відтворення. Саме тому, навіть за наявності потужних механізмів корекції помилок на канальному рівні, актуальними залишаються алгоритми відновлення відео на прикладному рівні, зокрема методи деконволюції. Підвищена ефективність LDPC зменшує кількість і масштаб втрат, а отже, створює сприятливіші умови для роботи деконволюційних алгоритмів, які відновлюють залишкові спотворення та підвищують загальну якість відеопотоку.

Таким чином, LDPC у DOCSIS 3.1 є ключовим елементом, що забезпечує високу пропускну здатність і стійкість каналу, але для запобігання прихованій деградації якості необхідне ретельне тестування системи та поєднання мережевих механізмів корекції з алгоритмічними методами відновлення відеоданих.

4.2 Максимальна пропускна здатність для запобігання втрат

Для успішного проведення тестування вкрай важливо чітко розуміти, з яких елементів складається OFDM і як вони взаємодіють між собою. Основою всієї системи є рівень PLC (PHY Link Channel), який містить службову інформацію, необхідну для коректного декодування OFDM-сигналу. Саме на цьому рівні

передаються параметри, що описують структуру сигналу, без яких кабельний модем не здатен виявити OFDM-несучу та визначити спосіб її декодування.

На вищому рівні розташовується вказівник на наступне кодове слово (Next Codeword Pointer, NCP). Його призначення полягає в інформуванні модему про те, яке кодове слово необхідно обробляти далі, а також який модуляційний профіль слід використовувати для декодування кожного конкретного кодового слова. Це забезпечує узгоджену роботу між передавачем і приймачем у динамічних умовах каналу.

Наступним елементом є профіль А, який виконує роль базового або завантажувального профілю. Кожен модем, що підтримує стандарт DOCSIS 3.1, зобов'язаний коректно працювати з цим профілем. Саме він дозволяє модему правильно інтерпретувати сигнал і забезпечує можливість переходу до використання вищих рівнів модуляції QAM, визначених у інших профілях. Таким чином, профіль А є фундаментом для стабільної роботи OFDM і подальшого застосування більш ефективних модуляційних схем.

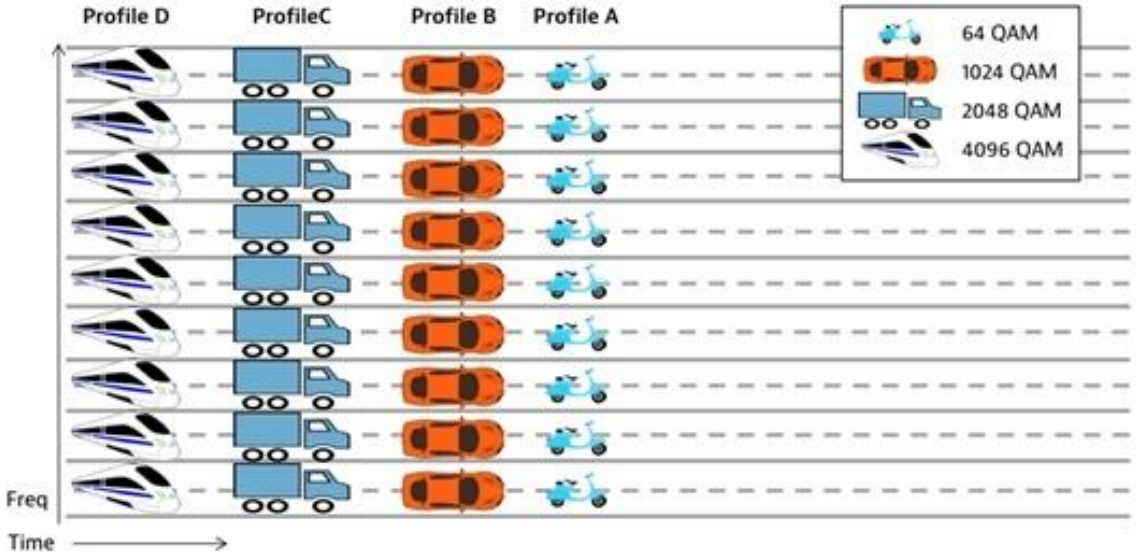


Рис. 4.3 Для спрощення приймемо, що профілі використовують однакову модуляцію всіх піднесучих

Параметри рівнів потужності сигналу, показника MER та рівня шуму, що використовуються в профілі А, спеціально підібрані таким чином, щоб забезпечити максимально надійну роботу OFDM навіть у несприятливих умовах кабельної мережі. Саме стабільна робота цього базового профілю є необхідною передумовою

для подальшого використання більш ефективних стандартних профілів B, C і D, які застосовуються вже для передавання основного трафіку з вищими рівнями модуляції. Окрім стандартних профілів, виробники CMTS і кабельних модемів мають можливість створювати власні профілі, кількість і склад яких формально не обмежуються специфікацією, що забезпечує додаткову гнучкість адаптації до конкретних умов мережі.

Під час передавання службової інформації рівня PLC критично важливо досягти повної відсутності некоректованих помилок кодових слів (uncorrectable codeword errors, CWE). Оскільки цей рівень відповідає за базову інформацію, необхідну для декодування OFDM-сигналу, його надійність має бути максимальною. Саме тому значення потужності сигналу та MER повинні перебувати в суворо визначених межах. Для забезпечення такої стабільності параметри PLC є жорстко фіксованими, а стандарт DOCSIS 3.1 дозволяє використовувати для цього рівня лише BPSK або 16 QAM, які є найбільш стійкими до завад і шумів.

Аналогічні вимоги висуваються і до параметрів NCP (Next Codeword Pointer). Якщо рівень PLC функціонує без помилок, то параметри NCP також фіксуються і не повинні допускати появи некоректованих CWE. Втрата повідомлень на цьому рівні може призвести до того, що модем буде змушений повторно запитувати службову інформацію, а в гіршому випадку — втратить зв’язок із мережею. З цієї причини в DOCSIS 3.1 для передавання NCP дозволено застосовувати лише відносно стійкі схеми модуляції: QPSK, 16 QAM або 64 QAM.

Оскільки профіль A виконує роль завантажувального та базового профілю, для нього зазвичай встановлюються нижчі рівні модуляції, ніж для інших профілів, зокрема 16 QAM та 64 QAM. Це робиться для того, щоб усі кабельні модеми могли коректно працювати навіть у найгірших ділянках кабельної мережі. Сигнали з нижчими рівнями модуляції здатні функціонувати при менших значеннях потужності та MER, що підвищує загальну надійність зв’язку.

Як і для рівнів PLC та NCP, параметри профілю A повинні залишатися фіксованими і не допускати появи некоректованих помилок кодових слів. Якщо такі

помилки все ж виникають, модем переходить у режим DOCSIS 3.0, унаслідок чого втрачається весь ефект підвищення спектральної ефективності, який забезпечує DOCSIS 3.1. Водночас профіль А може працювати і з більш високими рівнями модуляції, за умови що виникають лише коректовані помилки CWE, що вважається допустимим режимом роботи. Ключовою вимогою залишається відсутність некоректованих помилок, оскільки саме вони безпосередньо впливають на стабільність з'єднання та можливість подальшого використання високоефективних профілів передавання даних.

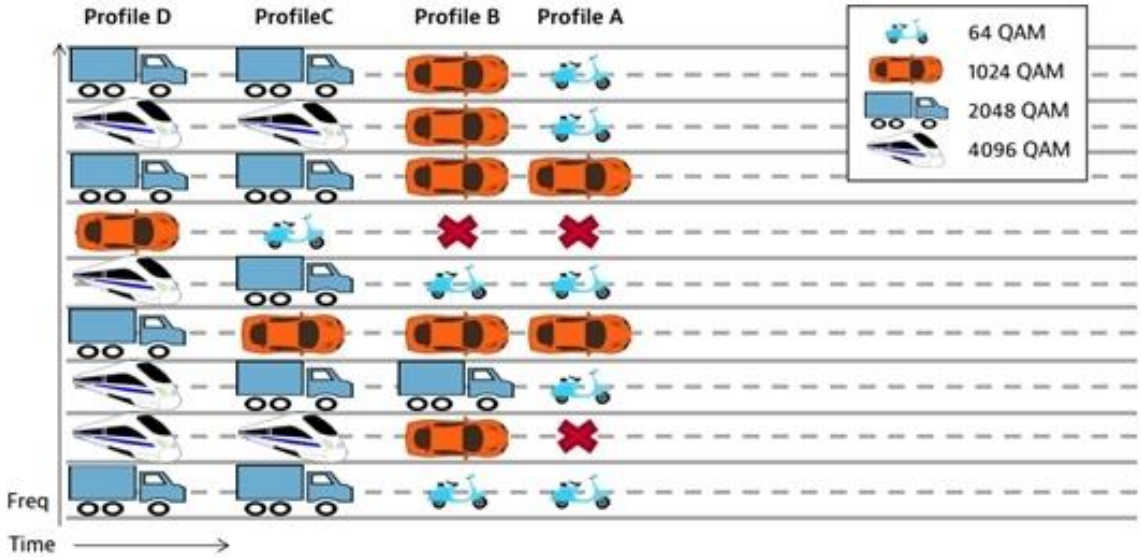


Рис. 4.4 OFDM дозволяє виключати певні піднесучі та дозволяє кожному мати різні рівні модуляції для різних піднесучих. Це оптимізує загальну пропускну спроможність каналу – кожен профіль має свої власні винятки

Коли всі три ключові рівні OFDM (PLC, NCP і профіль А) стабільно працюють у заданих межах, можна переходити до аналізу загальної пропускної здатності каналу. На цьому етапі часто допускається типова помилка — вимірювання рівня сигналу одразу по всій смузі 192 МГц без урахування особливостей розподілу потужності. Важливо пам'ятати, що сумарна потужність сигналу в широкій смузі OFDM не є безпосередньо співставною з потужністю окремої несучої шириною 6 МГц (або 8 МГц у європейських мережах). Фактично загальна потужність у смузі 192 МГц еквівалентна потужності одного 6-МГц сигналу з урахуванням масштабування за шириною смуги. Через це сумарний

рівень OFDM-сигналу може суттєво відрізнятися від рівня одиночної несучої, що потрібно обов’язково враховувати під час налаштувань.

Для більш точної й коректної конфігурації потужності OFDM усі вимірювання доцільно виконувати відносно еталонного сигналу шириною 6 МГц. Такий підхід дозволяє уникнути помилок інтерпретації показників і забезпечує коректне порівняння з традиційними SC-QAM каналами.

OFDM має й низку інших специфічних характеристик, які важливо враховувати під час аналізу. Зокрема, рівні перших і останніх 6 МГц у межах заданої OFDM-смуги зазвичай приблизно на 0,8 дБ нижчі, ніж рівні центральних піднесучих. Це пов’язано зі спадом амплітуди в захисних діапазонах (guard band). Така особливість набуває значення під час використання стандартних вимірювальних приладів або коли аналіз виконується у вузькому частотному вікні шириною 6 МГц, вирізаному з усього OFDM-спектра.

Крім того, несуча PLC зазвичай має рівень приблизно на 0,8 дБ вищий, ніж інші піднесучі. Це пояснюється наявністю додаткових пілотних сигналів і службових даних, які передаються на цьому рівні. У сукупності ці фактори призводять до того, що загальна рівномірність (flatness) OFDM-сигналу у порівнянні зі стандартним 6-МГц сигналом може коливатися в межах до 1,6 дБ, що зумовлено як крайовими спадами спектра, так і впливом PLC.

Для того щоб OFDM працював із піковою продуктивністю, середній рівень потужності сигналу повинен залишатися в заданих межах, значення MER має бути достатньо високим, а рівні шуму — мінімальними. Шум є одним із найбільш критичних факторів для OFDM, оскільки навіть незначне його зростання може призвести до того, що профілі з високими рівнями модуляції взагалі перестануть використовуватися системою адаптації.

За умови дотримання всіх зазначених вимог стає можливим застосування профілів із високими рівнями модуляції. При цьому принципово важливо, щоб параметри в межах кожного профілю були зафіксовані (locked). Профілі з високою модуляцією можуть допускати певну кількість коректованих помилок кодових слів (CWE) — це менш критично, ніж для базових рівнів. Однак поява некоректованих

помилок безпосередньо обмежує досягнення максимальної продуктивності. Наприклад, якщо в профілі C виникають некоректовані помилки, то профілі D і всі вищі не зможуть використовувати рівні модуляції, вищі за рівень профілю C.

Отже, для досягнення максимальної ефективності роботи OFDM і використання найвищих рівнів модуляції HFC-мережа повинна бути “чистою”, тобто не допускати появи некоректованих помилок. Це правило є справедливим не лише для DOCSIS 3.1, а й для попередніх версій стандарту, однак саме в умовах високої спектральної ефективності DOCSIS 3.1 вимоги до якості мережі стають особливо жорсткими.

4.3 Upstream

У стандарті DOCSIS 3.1 для організації зворотного каналу передавання даних застосовується технологія OFDMA (Orthogonal Frequency-Division Multiple Access). Вона є розвитком принципів OFDM і призначена для ефективного обслуговування множини абонентських пристроїв, які одночасно передають дані в напрямку до мережі оператора. OFDMA дозволяє гнучко розподіляти частотні ресурси між абонентами, призначаючи їм окремі набори піднесучих у частотній та часовій областях, що суттєво підвищує ефективність використання спектра та зменшує взаємні завади.

Однією з важливих особливостей OFDMA у DOCSIS 3.1 є можливість вибіркового відключення окремих піднесучих. Такий механізм використовується, зокрема, для забезпечення зворотної сумісності з існуючими каналами стандартів DOCSIS 2.0 та DOCSIS 3.0, які можуть працювати в тих самих частотних діапазонах. Завдяки цьому нове обладнання DOCSIS 3.1 може співіснувати з раніше впровадженими мережевими рішеннями без необхідності повної заміни інфраструктури, що спрощує поетапну модернізацію мережі та знижує витрати операторів.

Spectrum Usage Evolution Example					
		DOCSIS 3.0		DOCSIS 3.1	
	Parameter	Current	Stage 1	Stage 2	Stage 3
Downstream	Spectrum (MHz)	54 to 1002	108 to 1002	258 to 1218 with Amp upgrade	500 to 1794 with Tap upgrade
	Modulation	QAM-256	QAM-256	QAM-1024 and higher	QAM-1024 and higher
	Equivalent # of Channels	8	24	158	200
	Throughput (bps)	300M	1G	7G	10G+
Upstream	Spectrum (MHz)	5 to 42/65	5 to 85	5 to 204	5 to 400
	Modulation	QAM-64	QAM-64	QAM-256 and higher	QAM-1024 and higher
	Equivalent # of Channels	4	12	32	60
	Throughput (bps)	100M	300M	1G+	2G+

Рис. 4.5 Порівняльна таблиця DOCSIS 3.0 та DOCSIS 3.1

Можна сказати, що стандарт DOCSIS 3.1 суттєво підвищує спектральну ефективність і пропускну здатність кабельних мереж завдяки використанню OFDM/OFDMA та сучасних механізмів корекції помилок LDPC. Запровадження цих технологій дозволяє зменшити кількість втрат пакетів, підвищити стабільність каналу та ефективніше використовувати доступний частотний ресурс навіть за динамічних і несприятливих умов передавання. У контексті обробки мультимедійних даних це створює більш сприятливі мережеві умови для передавання відеопотоків і зменшує рівень їх деградації. Водночас залишкові втрати та спотворення, які неминуче виникають у мережах з обмеженою пропускною здатністю, можуть бути компенсовані на прикладному рівні за допомогою алгоритмів деконволюції, що виконують відновлення відеопотоків. Таким чином, DOCSIS 3.1 не лише підвищує ефективність транспортного рівня, а й опосередковано підсилює результативність алгоритмічних методів відновлення відео, забезпечуючи вищу якість сервісу для кінцевих користувачів.

ВИСНОВКИ

У даній кваліфікаційній роботі було досліджено алгоритми деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю з урахуванням як алгоритмічних, так і мережевих аспектів передавання даних. У ході роботи показано, що проблема деградації якості відео в сучасних мережах не може бути ефективно розв’язана виключно на одному рівні — ні лише за рахунок мережевих механізмів, ні лише за рахунок прикладної обробки.

Було встановлено, що мережі з обмеженою та динамічною пропускною здатністю, зокрема кабельні мережі доступу на основі стандарту DOCSIS 3.1, хоча й забезпечують суттєве підвищення спектральної ефективності завдяки використанню OFDM/OFDMA та алгоритмів корекції помилок LDPC, все одно залишаються схильними до втрат пакетів, коливань затримок і локальних спотворень відеопотоків. Навіть за наявності сучасних механізмів фізичного та канального рівнів повністю усунути вплив шумів, перевантажень і нестабільності каналу неможливо, особливо за пікових навантажень або у гетерогенних мережевих середовищах.

У цьому контексті алгоритми деконволюції відіграють ключову роль на прикладному рівні, забезпечуючи відновлення просторово-часової структури відео, компенсацію втрат фрагментів кадрів і зменшення візуальних артефактів. Показано, що ефективність деконволюційних моделей безпосередньо залежить від стабільності надходження відеоданих і супровідних метаданих, що підкреслює важливість оптимізації всієї інфраструктури передавання.

Окрему увагу в роботі було приділено потоковим архітектурам обробки даних, зокрема використанню Kafka та Kafka Connect як інфраструктурних компонентів для надійної доставки фрагментів відеопотоку. Продемонстровано, що правильне налаштування Source-конекторів і Kafka Producer (batch.size, linger.ms, batch.max.rows, poll.interval.ms тощо) дозволяє суттєво підвищити пропускну здатність системи, зменшити затримки та уникнути ситуацій, коли продюсер або конектор стають вузьким місцем. Це, у свою чергу, створює більш сприятливі умови

для подальшої роботи алгоритмів деконволюції, які потребують стабільного та впорядкованого потоку даних.

Таким чином, у роботі показано, що відновлення відеопотоків у мережах з обмеженою пропускною здатністю є комплексною задачею, яка вимагає поєднання сучасних мережевих технологій (таких як DOCSIS 3.1), оптимізованої потокової інфраструктури передавання даних і ефективних алгоритмів деконволюції. Саме інтеграція цих підходів дозволяє досягти підвищення якості відеосервісів і забезпечити прийнятний рівень якості обслуговування для кінцевих користувачів навіть за несприятливих мережевих умов.

ПЕРЕЛІК ПОСИЛАНЬ

1. Richardson W. H. Bayesian-Based Iterative Method of Image Restoration // Journal of the Optical Society of America. – 1972. – Vol. 62, № 1. – P. 55–59.
2. Lucy L. B. An Iterative Technique for the Rectification of Observed Distributions // The Astronomical Journal. – 1974. – Vol. 79. – P. 745–754.
3. Zeiler M. D., Krishnan D., Taylor G. W., Fergus R. Deconvolutional Networks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2010. – P. 2528–2535.
4. Wiener N. Extrapolation, Interpolation, and Smoothing of Stationary Time Series. – Cambridge : MIT Press, 1949. – 163 p.
5. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson Education, 2018. – 1168 p.
6. Szeliski R. Computer Vision: Algorithms and Applications. – 2nd ed. – Cham : Springer, 2022. – 1024 p.
7. Tekalp A. M. Digital Video Processing. – Upper Saddle River : Prentice Hall, 1995. – 650 p.
8. Wang Y., Zhu Q.-F. Error Control and Concealment for Video Communication: A Review // Proceedings of the IEEE. – 1998. – Vol. 86, № 5. – P. 974–997.
9. Sullivan G. J., Ohm J.-R., Han W.-J., Wiegand T. Overview of the High Efficiency Video Coding (HEVC) Standard // IEEE Transactions on Circuits and Systems for Video Technology. – 2012. – Vol. 22, № 12. – P. 1649–1668.
10. ITU-T Recommendation H.264. Advanced video coding for generic audiovisual services. – Geneva : International Telecommunication Union, 2019.
11. ITU-T Recommendation H.265. High efficiency video coding. – Geneva : International Telecommunication Union, 2021.
12. ISO/IEC 23009-1:2019. Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats.
13. Iyengar J., Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport : RFC 9000. – Internet Engineering Task Force, 2021.

14. CableLabs. DOCSIS® 3.1 Physical Layer Specification. – Louisville : Cable Television Laboratories, 2023.
15. CableLabs. DOCSIS® 3.1 MAC and Upper Layer Protocols Interface Specification. – Louisville : Cable Television Laboratories, 2023.
16. Apache Software Foundation. Apache Kafka Documentation. – Офіційна документація.
17. Confluent Inc. Kafka Connect and JDBC Source Connector Documentation. – Офіційна технічна документація.
18. Кирик О. В. Обмеження спотворень відеосигналів, спричинених втратою пакетів у телекомунікаційних мережах : дис. … канд. техн. наук. – Львів : Національний університет «Львівська політехніка», 2018. – 162 с.
19. Жураковський Б. Ю., Зєнів І. О. Комп’ютерні мережі : навч. посіб. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 312 с.
20. Бортник Г. Г. Системи доступу та передавання інформації в телекомунікаційних мережах : навч. посіб. – Вінниця : ВНТУ, 2012. – 268 с.
21. Мінухін С. В., Мірошник М. А. Технології мультимедійних даних : навч. посіб. – Харків : НТУ «ХПІ», 2018. – 240 с.
22. Ушенко Ю. О., Негрич А. Л., Галочкін О. В. Глобальна інформаційна інфраструктура : навч. посіб. – Київ : Центр навчальної літератури, 2022. – 286 с.
23. Системи і технології цифрового телебачення : навч. посіб. – Харків : ХНУРЕ, 2023. – 310 с.
24. Ахрамович С. М. Комп’ютерні мережі : навч. посіб. – Львів : Новий Світ–2000, 2017. – 344 с.
25. Поліщук О. С. Методи підвищення якості потокового відео в телекомунікаційних мережах з обмеженою пропускною здатністю // Вісник НТУУ «КПІ». – 2021. – № 3. – С. 45–52.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

КВАЛІФІКАЦІЙНА РОБОТА

АЛГОРИТМИ ДЕКОНВОЛЮЦІЇ ДЛЯ ВІДНОВЛЕННЯ ВІДЕОПОТОКІВ ПІСЛЯ ВТРАТ У МЕРЕЖАХ З ОБМЕЖЕНОЮ ПРОПУСКНОЮ ЗДАТНІСТЮ

Виконав:
студент групи ТСДМ-61
Андрій ЖИТКОВ

МЕТА РОБОТИ ТА АКТУАЛЬНІСТЬ

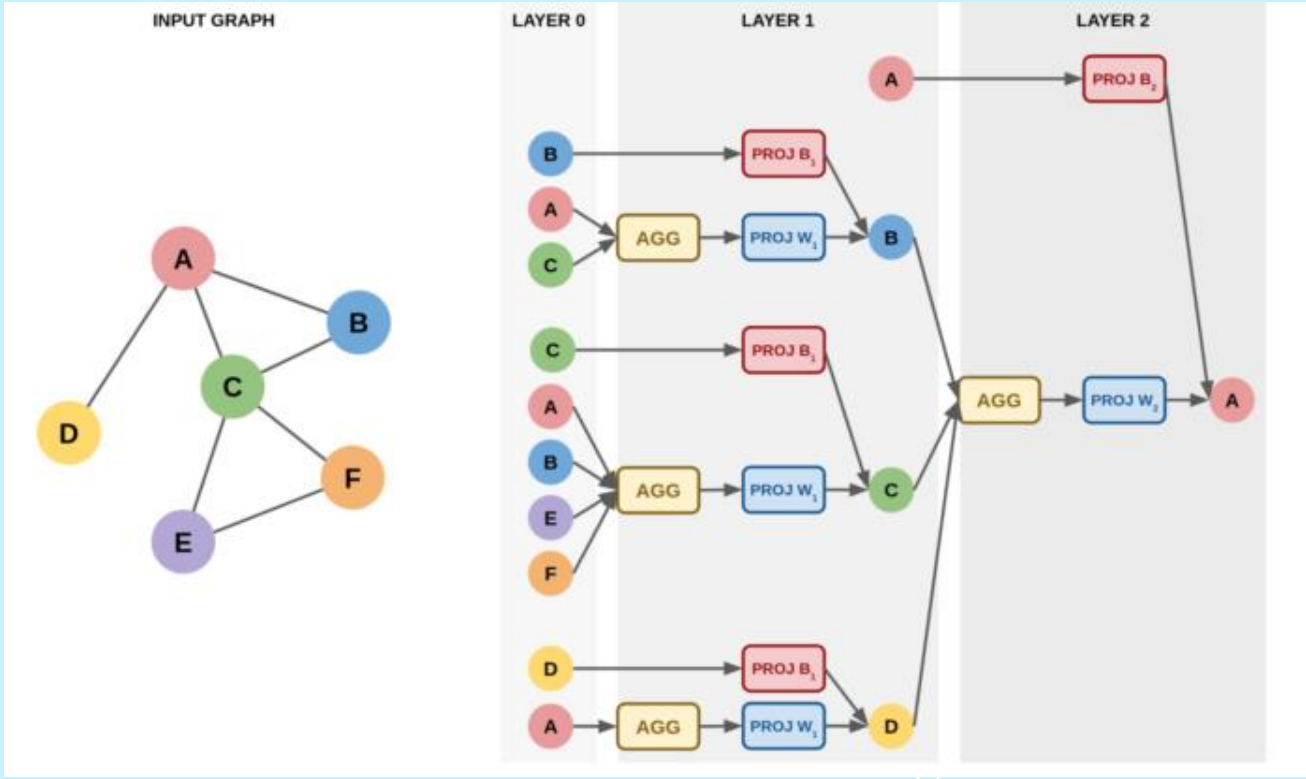
- Мета роботи – підвищення якості відновлення відеопотоків у телекомунікаційних мережах з обмеженою пропускною здатністю шляхом застосування алгоритмів деконволюції та оптимізації серверної й мережевої інфраструктури.
- Об’єкт дослідження – процес передавання та обробки відеопотоків у телекомунікаційних мережах з обмеженими та нестабільними параметрами пропускної здатності.
- Предмет дослідження – методи та алгоритми деконволюційного відновлення відеопотоків, а також способи оптимізації серверних і мережевих ресурсів, що впливають на якість відтворення відео після втрат даних.

Актуальність.

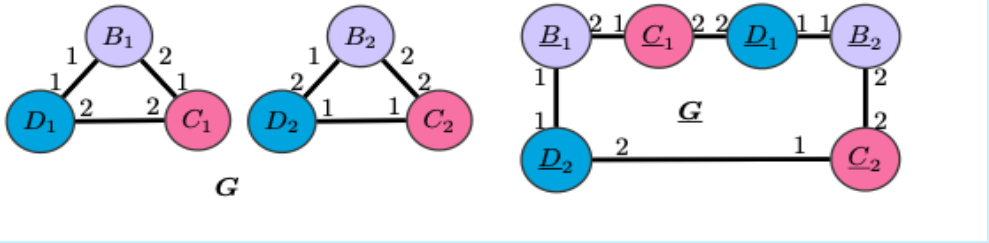
- Зростання обсягів відеотрафіку в сучасних телекомунікаційних мережах зумовлює підвищені вимоги до якості та надійності передавання даних. В умовах обмеженої пропускної здатності та нестабільних мережевих характеристик втрати пакетів і затримки призводять до суттєвої деградації якості відеопотоків. Традиційні мережеві механізми не завжди забезпечують необхідний рівень якості сервісу, що актуалізує використання алгоритмів деконволюції та інтелектуальної обробки відео на прикладному рівні. Поєднання таких методів з оптимізацією серверної та мережевої інфраструктури дозволяє підвищити стійкість відеопотоків і якість їх відтворення в реальних умовах експлуатації мереж.

ГРАФОВІ НЕЙРОННІ МЕРЕЖІ В ТЕЛЕКОМУНІКАЦІЯХ

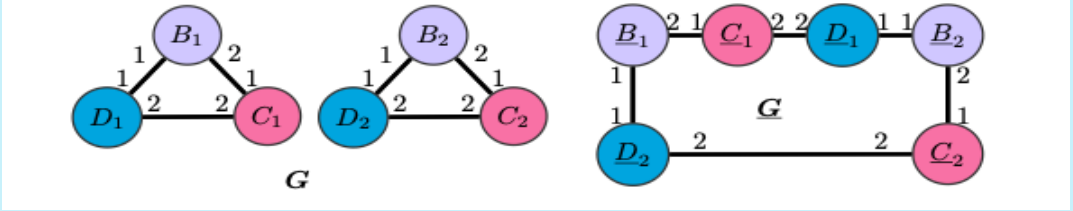
- Графові нейронні мережі (GNN) — це моделі машинного навчання, призначені для роботи з графами, які враховують як ознаки вершин, так і топологію зв'язків між ними. Оновлення представлень вершин здійснюється шляхом агрегації інформації від сусідів, що інтерпретується як згортка на графі, а деконволюційні механізми можуть використовуватися для відновлення або поширення узагальнених ознак. Найпоширенішими архітектурами є GCN, GraphSAGE, GAT та GIN.
- У задачах класифікації графів спочатку формуються ембеддинги вершин, які згодом агрегуються в єдине представлення всього графа за допомогою інваріантних функцій. Подібні принципи застосовуються й у телекомунікаційних задачах, зокрема для відновлення відеопотоків.
- Водночас GNN мають обмеження: локально інваріантні моделі втрачають частину структурної інформації, підходи з упорядкуванням сусідів залежать від нумерації, а геометричні моделі типу DimeNet мають високу обчислювальну складність і обмежену універсальність.



Дерево обчислень для двошарової GNN з прикладу вузла A



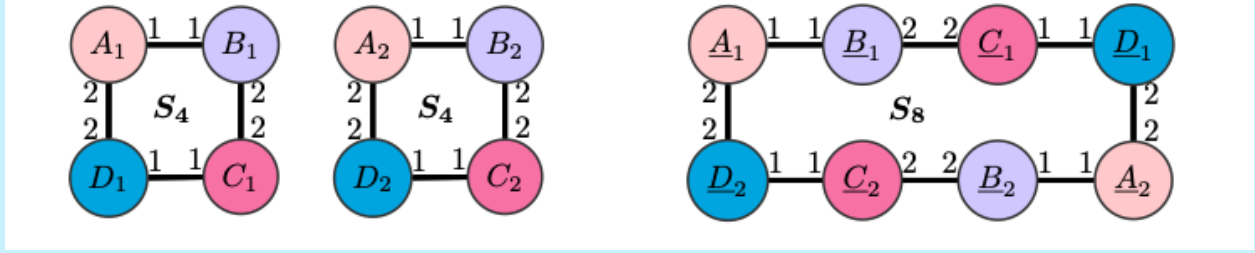
Приклад графа G з різними варіантами локального упорядкування вершин і ребер (port numbering), що демонструє неоднозначність представлення структури при однаковій топології



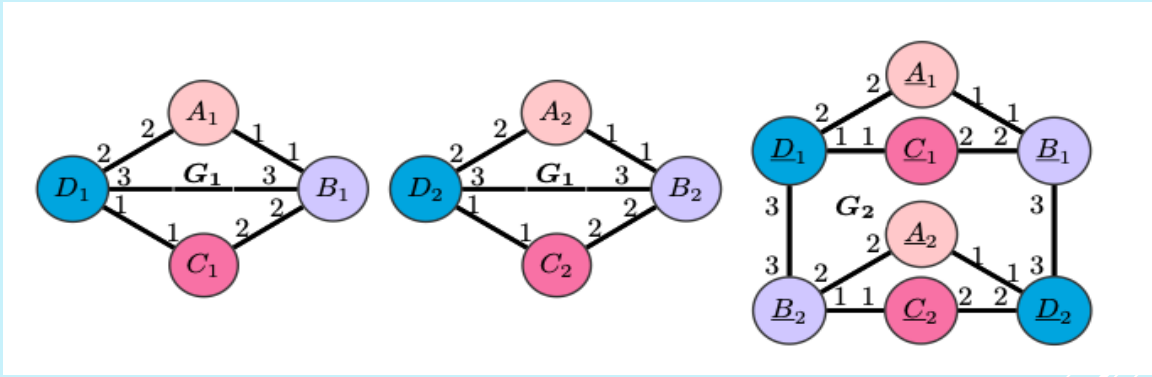
Ілюстрація двох ізоморфних графів GGG з однаковою топологією, але різним локальним упорядкуванням сусідніх вершин (port numbering), що впливає на здатність графових нейронних мереж розрізняти їх структуру

- Локально інваріантні GNN (LU-GNN) не розрізняють різні графи з однаковими локальними структурами. CPNGNN підвищують розрізнювальну здатність завдяки локальному упорядкуванню, але втрачають глобальні топологічні відмінності через інваріантний readout. DimeNet враховує геометрію, однак і він не розрізняє графи з однаковими відстанями та кутами. Це демонструє фундаментальні обмеження сучасних графових нейронних мереж.

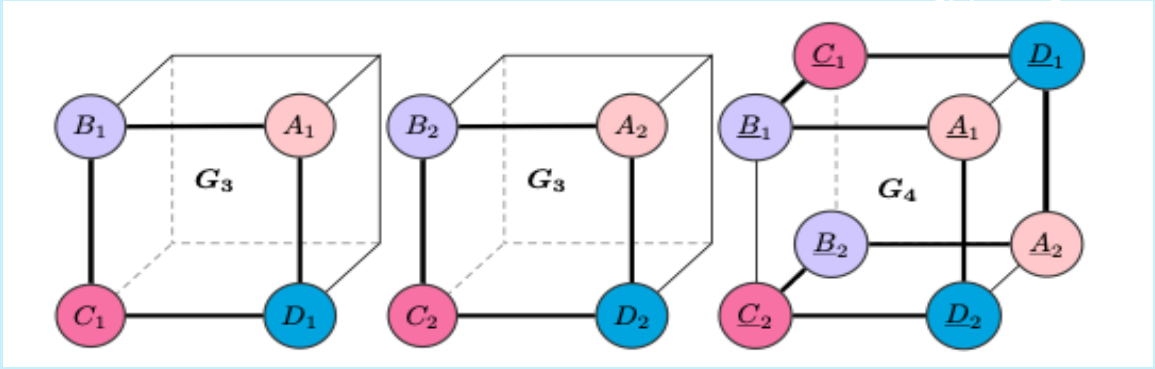
ОБМЕЖЕННЯ В ГРАФОВИХ СИСТЕМАХ



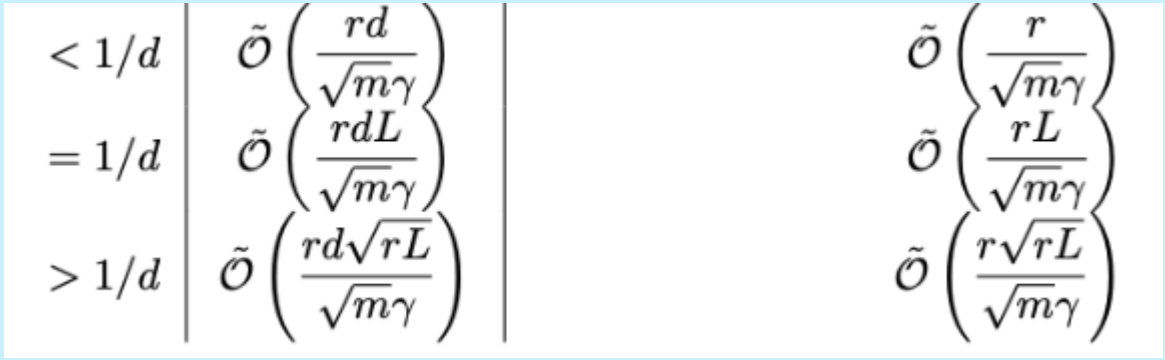
Приклад графів S_4 та S_8, що демонструють вплив локальної симетрії та нумерації портів на розрізнювальну здатність графових нейронних мереж при аналізі вершин і структури графа



Приклад графів G_1 та G_2 з різною глобальною структурою та кількістю циклів, які, попри топологічні відмінності, можуть мати однакові ембеддинги при використанні CPNGNN з фіксованим порядком портів і readout-функцією, інваріантною до перестановок



Приклад графів G_3 та G_4 з різною просторовою конфігурацією вершин, що ілюструє вплив геометричної структури на розрізнювальну здатність графових нейронних мереж, зокрема при врахуванні просторових залежностей і кутів між ребрами



$< 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd}{\sqrt{m\gamma}}\right)$	$\tilde{\mathcal{O}}\left(\frac{r}{\sqrt{m\gamma}}\right)$
$= 1/d$	$\tilde{\mathcal{O}}\left(\frac{rdL}{\sqrt{m\gamma}}\right)$	$\tilde{\mathcal{O}}\left(\frac{rL}{\sqrt{m\gamma}}\right)$
$> 1/d$	$\tilde{\mathcal{O}}\left(\frac{rd\sqrt{rL}}{\sqrt{m\gamma}}\right)$	$\tilde{\mathcal{O}}\left(\frac{r\sqrt{rL}}{\sqrt{m\gamma}}\right)$

Порівняння з RNN

КРОК ДО БІЛЬШ ПОТУЖНИХ GNN. ОЦІНКА ПОМИЛКИ УЗАГАЛЬНЕННЯ

- Запропоновано об’єднати існуючі підходи GNN та доповнити їх розширеним набором геометричних характеристик з метою підвищення розрізнювальної здатності моделей. Окрім відстаней і кутів між ребрами, враховуються кути між площинами, заданими трійками ребер, що дозволяє моделювати геометричні залежності вищих порядків і точніше відображати глобальну структуру графа.
- Запропонована GNN функціонує так: на першому етапі застосовується шар DimeNet для базового урахування геометрії. Далі message-ембеддинг ребра доповнюється геометричними ознаками

$$m_{wv}^{(l)} = f!\left(m_{wv}^{(l)}, \Phi_{wv}\right)$$

а представлення вершини обчислюється з урахуванням локального порядку сусідів

$h_v^{(l+1)} = f\left(h_v^{(l)}, m_{c_v(1),v}^{(l)}, t_{1,v}, \ldots, m_{c_v(d(v)),v}^{(l)}, t_{d(v),v}\right)$

після чого використовується класифікатор або регресор.

- Для оцінки узагальнення розглядається клас LU-GNN з оновленням

$$h_v^{t+1} = \phi!\left(W_1 x_v + W_2 \rho!\left(\sum_{u \in \mathcal{N}(v)} g(h_u^t)\right)\right)$$

де функції активації є ліпшицевими, а норми ваг обмежені. Якість прогнозу описується величиною $p(f(G), y) = y(2f(G) - 1) + (1 - y)(1 - 2f(G))$, *а емпірична помилка визначається як* $\hat{\mathcal{R}}(f) = \frac{1}{m}\sum_{j=1}^{m} \text{loss}_\gamma!\left(-p!\left(f(G_j), y_j\right)\right)$.

- Показано, що з високою ймовірністю помилка узагальнення обмежується сумою помилки на тренувальній вибірці та складової, пов’язаної з виразною здатністю моделі. Аналіз зводиться до вивчення обчислювальних дерев локальних околиць вершин, що дозволяє довести стабільність GNN до малих змін параметрів. Отримані оцінки є точнішими за попередні та релевантні не лише для класифікації графів, а й для задач відновлення відеопотоків у мережах з обмеженою пропускною здатністю

```
usecs               : count     distribution
    0 -> 1          : 819      |                                        |
    2 -> 3          : 58888    |*****************************           |
    4 -> 7          : 77984    |****************************************|
    8 -> 15         : 10529    |*****                                   |
   16 -> 31         : 4853     |**                                      |
   ...
 4096 -> 8191       : 34       |                                        |
 8192 -> 16383      : 39       |                                        |
16384 -> 32767      : 17       |                                        |
```

Відстежування за допомогою запуску runqlat на одному з PID nginx-воркера

```
$ numastat -n -c
                Node 0   Node 1 Node 2 Node 3    Total
              -------- -------- ------ ------ --------
Numa_Hit      26833500 11885723      0      0 38719223
Numa_Miss        18672  8561876      0      0  8580548
Numa_Foreign   8561876    18672      0      0  8580548
Interleave_Hit  392066   553771      0      0   945836
Local_Node     8222745 11507968      0      0 19730712
Other_Node     18629427  8939632      0      0 27569060
```

Перевірка за допомогою numastat

```
$ numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 16 17 18 19
node 0 size: 32149 MB
node 1 cpus: 4 5 6 7 20 21 22 23
node 1 size: 32213 MB
node 2 cpus: 8 9 10 11 24 25 26 27
node 2 size: 0 MB
node 3 cpus: 12 13 14 15 28 29 30 31
node 3 size: 0 MB
node distances:
node   0   1   2   3
  0:  10  16  16  16
  1:  16  10  16  16
  2:  16  16  10  16
  3:  16  16  16  10
```

Перевірка топології за допомогою numactl --hardware

```
# lspci -s 0a:00.0 -vvv
...
LnkCap: Port #0, Speed 8GT/s, Width x8, ASPM L1, Exit Latency L0s <2us, L1 <16us
LnkSta: Speed 8GT/s, Width x8, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
...
Capabilities: [100 v2] Advanced Error Reporting
UESta:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- ...
UEMsk:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- ...
UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- ...
CESta:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr-
CEMsk:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
```

Попередній моніторинг за допомогою утиліти lspci

```
$ numastat -m -c
                Node 0 Node 1 Node 2 Node 3 Total
              ------ ------ ------ ------ -----
MemTotal       32150  32214      0      0 64363
MemFree          462   5793      0      0  6255
MemUsed        31688  26421      0      0 58109
Active         16021   8588      0      0 24608
Inactive       13436  16121      0      0 29557
Active(anon)    1193    970      0      0  2163
Inactive(anon)   121    108      0      0   229
Active(file)   14828   7618      0      0 22446
Inactive(file) 13315  16013      0      0 29327
...
FilePages      28498  23957      0      0 52454
Mapped           131    130      0      0   261
AnonPages        962    757      0      0  1718
Shmem            355    323      0      0   678
KernelStack       10      5      0      0    16
```

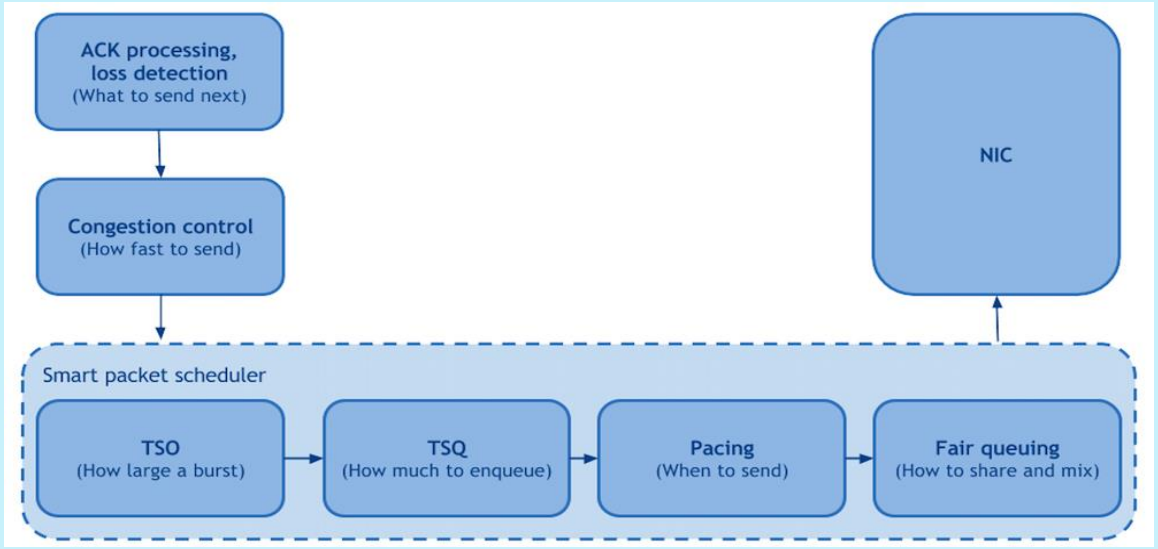
Статистика використання пам’яті

ОПТИМІЗАЦІЯ ДЛЯ ПІДВИЩЕННЯ ПРОПУСКНОЇ ЗДАТНОСТІ ТА ЗМЕНШЕННЯ ЗАТРИМКИ

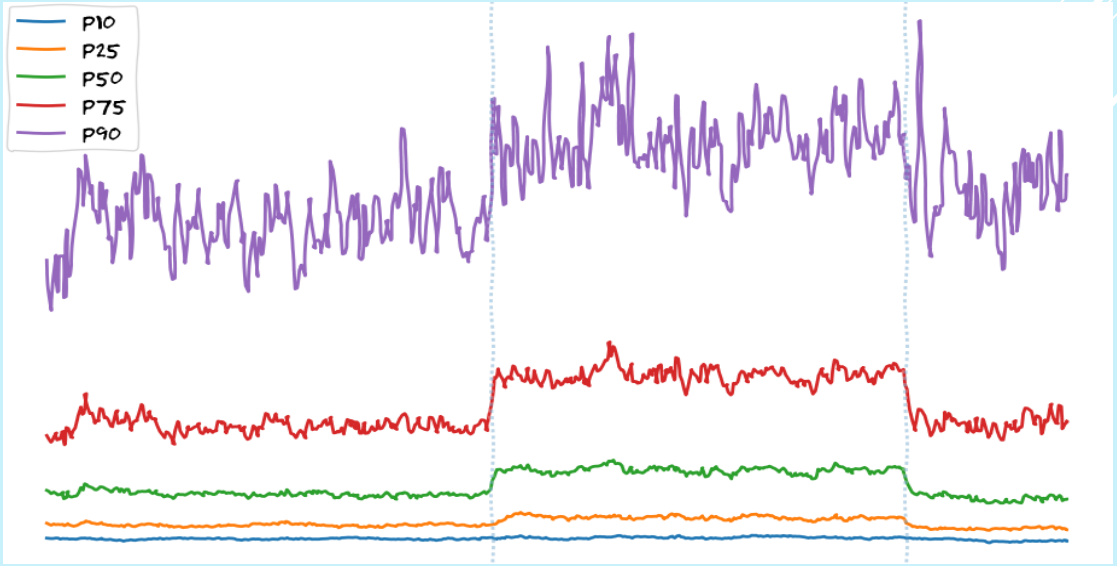
- Оптимізація високонавантажених систем виконується на кількох рівнях: апаратному, рівні операційної системи та прикладному. Базові налаштування обладнання, зокрема процесора, мережевих адаптерів і пам’яті, визначають граничну продуктивність і затримки. На рівні ОС ключову роль відіграє сучасний мережевий стек Linux із механізмами керування чергами, перевантаженнями та NUMA-локальністю. Прикладні оптимізації, зокрема буферизація, асинхронна обробка та компресія, дозволяють ефективніше використовувати ресурси. Сукупно ці заходи зменшують втрати пакетів і затримки, створюючи сприятливі умови для якісного відновлення відеопотоків у мережах з обмеженою пропускною здатністю.

ОГЛЯД ВІДНОВЛЮВАЛЬНИХ АЛГОРИТМІВ

- Документ *Making Linux TCP Fast* систематизує ключові оптимізації TCP-стеку Linux, що накопичувалися протягом багатьох років, і розглядає його як сукупність взаємопов'язаних функціональних механізмів. До них належать fair queueing і pacing, які забезпечують рівномірний розподіл трафіку між потоками, зменшують перевантаження черг і знижують втрати пакетів, а також автоматичне керування TSO та TSQ, спрямоване на зменшення затримок без втрати пропускної здатності.
- Суттєвий внесок у підвищення ефективності роблять сучасні алгоритми керування перевантаженнями, зокрема BBR, які орієнтуються на затримку, а не лише на втрати пакетів. У поєднанні з оновленими механізмами обробки ACK і виявлення втрат ці підходи дозволяють стабілізувати роботу TCP у мережах зі змінною та обмеженою пропускною здатністю й підвищити якість доставки мультимедійних даних.



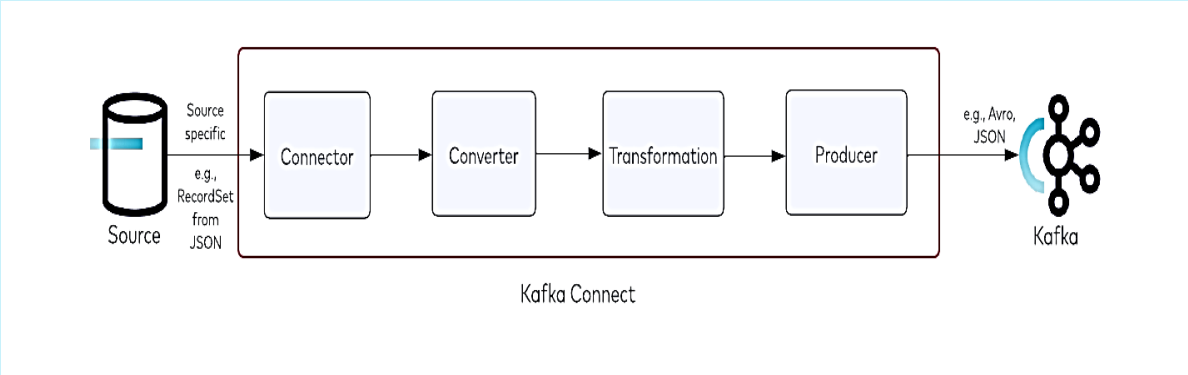
Функціональна схема TCP-стеку на стороні відправника в Linux, що ілюструє послідовність обробки ACK і виявлення втрат, керування перевантаженням, інтелектуальне планування пакетів (TSO, TSQ, pacing, fair queuing) та передачу даних на мережевий адаптер (NIC)



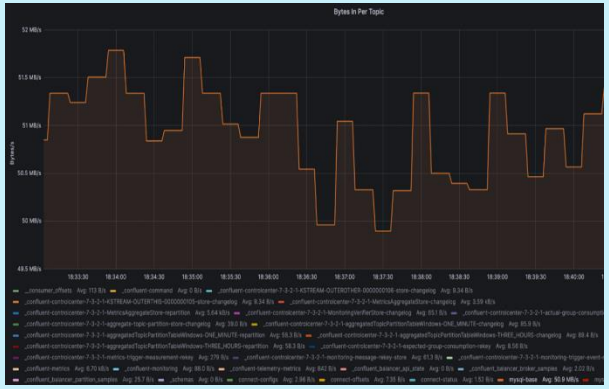
Шестигодинний експеримент із TCP BBR у Tokyo PoP: вісь x — час, вісь y — швидкість завантаження на клієнта

ВИКОРИСТАННЯ ІНФРАСТРУКТУРНОГО КОМПОНЕНТУ KAFAKA CONNECT

- Kafka Connect використовується як інфраструктурний компонент потокової обробки даних, що забезпечує надійну та масштабовану доставку фрагментів відеопотоку разом із супровідними метаданими. Завдяки механізмам буферизації, повторної доставки та збереження порядку повідомлень він створює стабільне середовище для роботи алгоритмів деконволюції, які виконують відновлення відео на прикладному рівні.
- Архітектура Source-конектора включає джерело даних, конектор, конвертер, трансформації та Kafka Producer. Частина компонентів (конвертери й трансформації) має неминучі накладні витрати, тоді як продуктивність системи може бути оптимізована за рахунок налаштувань конектора та продюсера. Для конектора ключовими параметрами є розмір пакета зчитування та інтервал опитування джерела, а для продюсера — параметри пакетування, буферизації та стиснення повідомлень.
- Ефективне налаштування Kafka Connect полягає у збалансованому підборі цих параметрів з урахуванням характеристик джерела даних, вимог до пропускної здатності та допустимих затримок, що дозволяє підвищити загальну продуктивність потокової системи без модифікації її програмної логіки.



Архітектура Source-конектора в Kafka Connect



BytesInPerSec - 50.9 MB/s

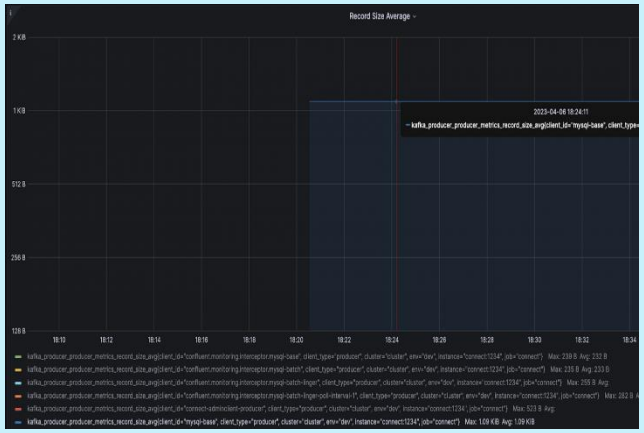


source-record-poll-rate - 53.1K в секунду

Рівень	Метрика	Опис
Коннектор	source-record-poll-rate	Середня кількість записів за секунду, які задача (task) Source-конектора зчитує з джерела або генерує до застосування трансформацій у межах робочого процесу (worker).
Коннектор	poll-batch-avg-time-ms	Середній час у мілісекундах, який задача витрачає на отримання одного пакета записів від джерела.
Коннектор	source-record-write-rate	Середня кількість записів за секунду, які після застосування трансформацій записуються в Kafka (без урахування записів, відфільтрованих SMT).
Брокер	kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec	Швидкість надходження байтів від клієнтів до брокера Kafka для конкретного топіка.
Продюсер	record-size-avg	Середній розмір одного запису, який відправляється продюсером.
Продюсер	batch-size-avg	Середня кількість байтів, що відправляються у партицію за один запит продюсера.
Продюсер	records-per-request-avg	Середня кількість записів, які надсилаються в одному запиті продюсера.
Продюсер	record-send-rate	Середня кількість записів, що надсилаються продюсером у топік за секунду.

Оновлені дані метрик

Метрика	Значення
BytesInPerSec	50.9 MB/s
source-record-poll-rate	53.1K ops/s
record-size-avg	1.09 KiB



record-size-avg - 1.09 KiB

```
curl -X PUT \
    -H "Content-Type: application/json" \
    --data '{
            "connector.class":"io.confluent.connect.jdbc.JdbcSourceConnector",
            "tasks.max":"1",
            "connection.url":"jdbc:mysql://mysql:3306/mydb?user=user&password=password&use
            "mode":"bulk",
            "batch.max.rows": 500,
            "producer.override.client.id": "mysql-batch",
            "producer.override.batch.size": 558080,
            "query":"SELECT * FROM mydb.team WHERE mydb.team.id < 7900000 ",
            "topic.prefix":"mysql-batch"
        }' \
```

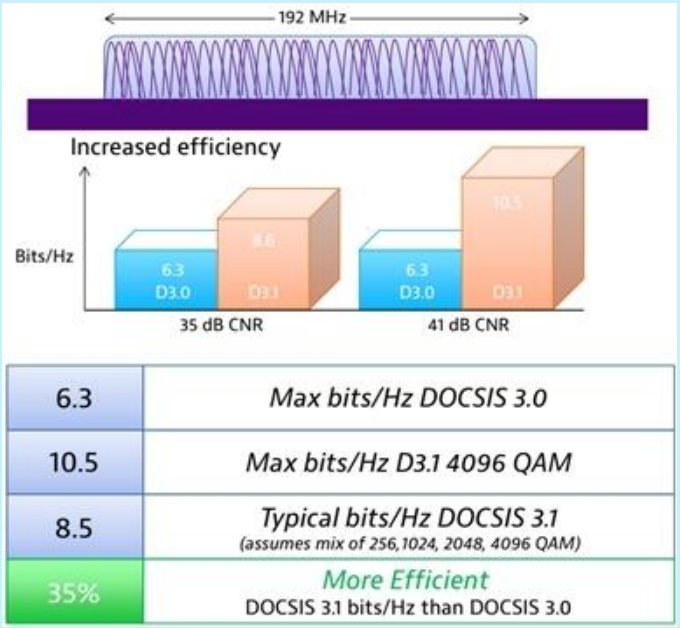
Параметри batch.max.rows та batch.size збільшилися

JMX-метрики Kafka Connect доцільно поділяти на метрики конектора, продюсера та брокера, що дозволяє системно аналізувати продуктивність потокової інфраструктури та локалізувати вузькі місця. Поетапний підхід передбачає фіксацію базових показників, аналіз ключових метрик і визначення рівня, на якому оптимізація буде найбільш ефективною. У розглянутому сценарії первинним обмеженням виявився Kafka Producer через недостатній розмір пакетів, після усунення якого вузьким місцем став конектор. Подальша оптимізація параметрів конектора, зокрема частоти опитування джерела, дозволяє підвищити загальну пропускну здатність системи та стабільність передавання даних.

JMX МЕТРИКИ

- Стандарт DOCSIS 3.1 визначає механізми передавання даних у кабельних мережах з обмеженою та динамічною пропускною здатністю, де можливі втрати пакетів, коливання затримок і нестабільність швидкості. Завдяки застосуванню технологій OFDM та алгоритмів корекції помилок LDPC досягається суттєве підвищення спектральної ефективності, пропускної здатності та стійкості каналу порівняно з DOCSIS 3.0.
- Водночас навіть за наявності вдосконалених механізмів фізичного та канального рівнів у реальних умовах залишаються спотворення відеопотоків, особливо за пікових навантажень. Тому актуальним є використання алгоритмів деконволюції на прикладному рівні, які компенсують залишкові втрати та підвищують якість відновлення відео, забезпечуючи кращий рівень якості обслуговування користувачів.

ПІДВИЩЕННЯ ПРОПУСКНОЇ ЗДАТНОСТІ ДЛЯ УСУНЕННЯ ВТРАТ В МЕРЕЖІ



6.3	*Max bits/Hz DOCSIS 3.0*
10.5	*Max bits/Hz D3.1 4096 QAM*
8.5	*Typical bits/Hz DOCSIS 3.1* (assumes mix of 256,1024, 2048, 4096 QAM)
35%	*More Efficient* DOCSIS 3.1 bits/Hz than DOCSIS 3.0

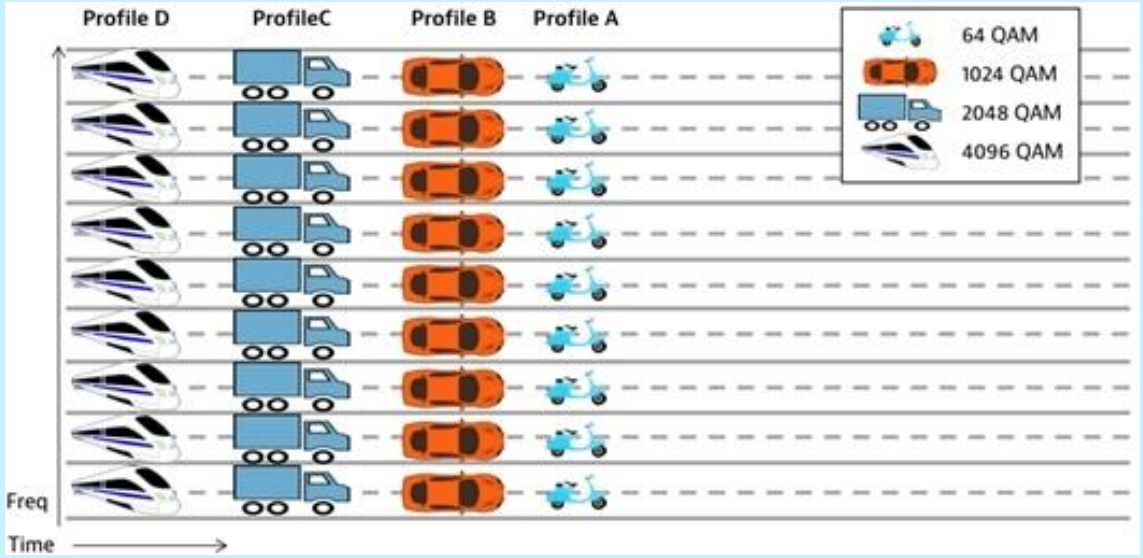
Порівняння спектральної ефективності стандартів DOCSIS 3.0 та DOCSIS 3.1, що ілюструє збільшення кількості переданих бітів на 1 Гц смуги частот завдяки використанню OFDM і високорівневої модуляції (до 4096-QAM) у DOCSIS 3.1



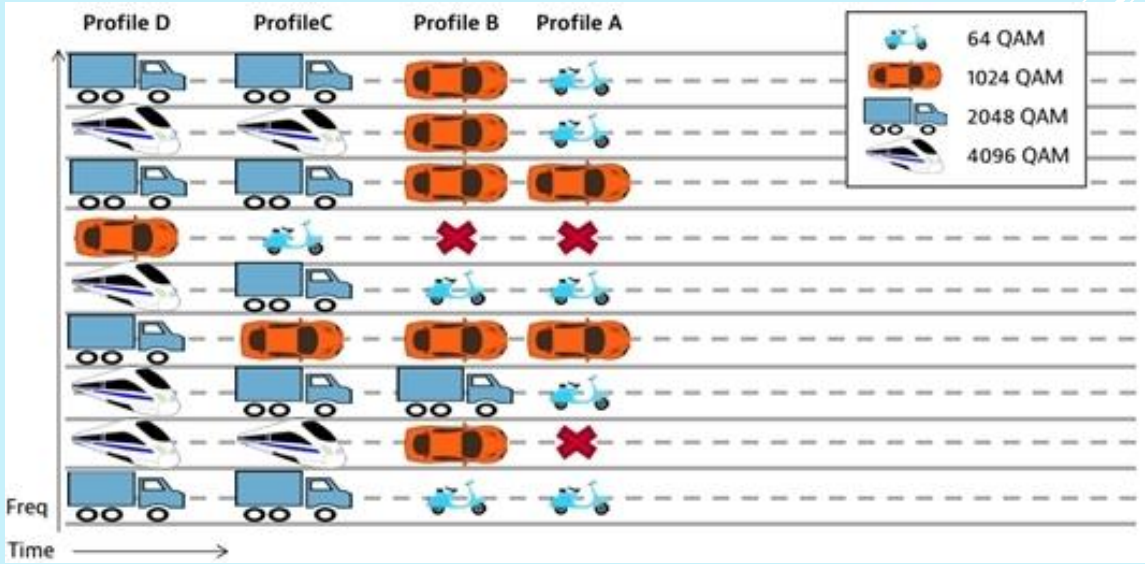
Порівняння традиційної модуляції SC-QAM та ортогонального частотного мультиплексування OFDM, що ілюструє розподіл символів у часі та частоті, а також гнучке використання піднесучих і різних рівнів QAM у DOCSIS 3.1

- Для коректної роботи OFDM у стандарті DOCSIS 3.1 ключовими є рівні PLC, NCP та базовий профіль A, які забезпечують декодування сигналу й узгоджену роботу модемів. Ці рівні мають функціонувати без некоректованих помилок і використовують стійкі схеми модуляції. Профіль A створює основу для застосування більш ефективних профілів з вищою модуляцією. Ефективність OFDM залежить від коректного налаштування потужності, високого MER і низького рівня шуму, оскільки некоректовані помилки безпосередньо обмежують пропускну здатність і можливість використання високих рівнів модуляції.

МАКСИМАЛЬНА ПРОПУСКНА ЗДАТНІСТЬ ДЛЯ ЗАПОБІГАННЯ ВТРАТ



Для спрощення приймемо, що профілі використовують однакову модуляцію всіх піднесучих



OFDM дозволяє виключати певні піднесучі та дозволяє кожному мати різні рівні модуляції для різних піднесучих. Це оптимізує загальну пропускну спроможність каналу – кожен профіль має свої власні винятки

UPSTREAM

- У стандарті DOCSIS 3.1 для організації зворотного каналу передавання даних використовується технологія OFDMA, яка забезпечує ефективне обслуговування множини абонентських пристроїв за рахунок гнучкого розподілу частотних і часових ресурсів. OFDMA дозволяє підвищити спектральну ефективність і зменшити взаємні завади між абонентами, а також підтримує вибіркове відключення піднесучих для забезпечення зворотної сумісності з мережами DOCSIS 2.0 і 3.0, що спрощує поетапну модернізацію інфраструктури.
- Застосування OFDM/OFDMA разом із сучасними механізмами корекції помилок LDPC у DOCSIS 3.1 підвищує пропускну здатність і стабільність кабельних мереж, зменшуючи втрати пакетів і деградацію відеопотоків. Водночас залишкові спотворення в мережах з обмеженою пропускною здатністю можуть компенсуватися на прикладному рівні за допомогою алгоритмів деконволюції, що забезпечує додаткове підвищення якості відеосервісів.

Spectrum Usage Evolution Example					
		DOCSIS 3.0		DOCSIS 3.1	
	Parameter	Current	Stage 1	Stage 2	Stage 3
Downstream	Spectrum (MHz)	54 to 1002	108 to 1002	258 to 1218 with Amp upgrade	500 to 1794 with Tap upgrade
	Modulation	QAM-256	QAM-256	QAM-1024 and higher	QAM-1024 and higher
	Equivalent # of Channels	8	24	158	200
	Throughput (bps)	300M	1G	7G	10G+
Upstream	Spectrum (MHz)	5 to 42/65	5 to 85	5 to 204	5 to 400
	Modulation	QAM-64	QAM-64	QAM-256 and higher	QAM-1024 and higher
	Equivalent # of Channels	4	12	32	60
	Throughput (bps)	100M	300M	1G+	2G+

Порівняльна таблиця DOCSIS 3.0 та DOCSIS 3.1

ВИСНОВКИ

- У кваліфікаційній роботі досліджено алгоритми деконволюції для відновлення відеопотоків після втрат у мережах з обмеженою пропускною здатністю з урахуванням як алгоритмічних, так і мережевих аспектів передавання даних.
- Показано, що деградація якості відео в сучасних мережах не може бути ефективно усунена виключно на одному рівні та потребує комплексного підходу.
- Встановлено, що навіть у мережах доступу на основі стандарту DOCSIS 3.1, які використовують OFDM/OFDMA та LDPC, залишаються втрати пакетів і коливання затримок.
- Обґрунтовано роль алгоритмів деконволюції у відновленні просторово-часової структури відео та зменшенні візуальних артефактів.
- Окрему увагу приділено оптимізації потокової інфраструктури передавання даних на основі Kafka, що забезпечує стабільне надходження відеоданих і підвищує ефективність їх подальшої обробки.

ДЯКУЮ ЗА УВАГУ!

Протокол аналізу звіту подібності науковим керівником

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

Автор: Андрій ЖИТКОВ

Назва: АЛГОРИТМИ ДЕКОНВОЛЮЦІЇ ДЛЯ ВІДНОВЛЕННЯ ВІДЕОПОТОКІВ ПІСЛЯ ВТРАТ У МЕРЕЖАХ З ОБМЕЖЕНОЮ ПРОПУСКНОЮ ЗДАТНІСТЮ

Координатор: Олександр КРИВОНОСЕНКО

Підрозділ: ННІТ

Коефіцієнт подібності 1:0.4

Коефіцієнт подібності 2:0

Тривога: 8

Після аналізу Звіту подібності констатую наступне:

☐ виявлені в роботі запозичення є сумлінними і не мають ознак плагіату. Тому робота визнається самостійною і допускається до захисту;

☐ виявлені в роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на доопрацювання;

☐ виявлені в роботі запозичення є недобросовісними і мають ознаки плагіату або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. У зв'язку з чим, робота не допускається до захисту.

…… ……

Дата *Підпис Наукового керівника*