МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

Петровський Юрій Миколайович

Програмний модуль аналізу відеопотоків з використанням LLM мереж/ A software module for analyzing video streams using LLM-based networks

спеціальність: 123 – Комп’ютерна інженерія
освітньо-професійна програма – Комп’ютерна інженерія

Випускна кваліфікаційна робота

Виконав: студент групи КІ-41
Петровський Юрій Миколайович

Науковий керівник
О.Й. Піцун

2025

АНОТАЦІЯ

Петровський Ю.М. Програмний модуль аналізу відеопотоків з використанням LLM мереж. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 «Комп’ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2025.

Робота написана обсягом 54 сторінку і містить 9 рисунків 29 джерела за переліком посилань

Мета дослідження - створити систему для моніторингу потоків OTT. Використовуються алгоритми штучного інтелекту щоб знайти та проаналізувати проблеми якості відео на рівні мережевих протоколів в реальному часі.

Об'єктом дослідження є процеси спостереження мережі та контроль якості потоків відео по OTT в розподілених телекомунікаційних системах.

Предметом дослідження є архітектурні принципи та програмні рішення для створення інтелектуальних систем моніторингу якості OTT-потоків з використанням гібридного підходу, що інтегрує комп'ютерний зір, аналіз мережевих метрик та обробку природної мови для автоматизації процесів технічного контролю медіаконтенту.

Практична цінність дослідження полягає в створенні готового до продажу програмного продукту який може сильно покращити обслуговування користувачів OTT-платформ і зменшити витрати провайдерів через автоматизацію процесів моніторингу мережі та аналізу якості медіа.

Ключові слова: OTT, ВІДЕОСТРІМІНГ, МОНІТОРИНГ, ШТУЧНИЙ ІНТЕЛЕКТ, ЯКІСТЬ МЕДІАКОНТЕНТУ

ANNOTATION

Petrovskyi Yu.M. A software module for analyzing video streams using LLM-based networks. – Manuscript.

Qualification thesis for obtaining the Bachelor's degree in specialty 123 Computer Engineering, educational-professional program. West Ukrainian National University, Ternopil, 2025.

The thesis comprises 54 pages, includes 9 figures and 29 references.

The aim of the research is to develop a system for monitoring OTT streams. Artificial intelligence algorithms are applied to detect and analyze video quality issues at the level of network protocols in real time.

The object of the research is the processes of network monitoring and quality control of video streams over OTT in distributed telecommunication systems.

The subject of the research is the architectural principles and software solutions for creating intelligent systems for OTT stream quality monitoring using a hybrid approach that integrates computer vision, network metrics analysis, and natural language processing to automate the processes of technical control of media content.

The practical value of the research lies in the development of a market-ready software product that can significantly improve the service quality for OTT platform users and reduce providers' costs through the automation of network monitoring and media quality analysis processes.

Keywords: OTT, VIDEO STREAMING, MONITORING, ARTIFICIAL INTELLIGENCE, MEDIA CONTENT QUALITY.

ЗМІСТ

ВСТУП

Індустрія відеостримінгу росте дуже швидко, що ставить серйозні труднощі для мережевих інженерів і архітекторів розподілених систем. Протягом останніх п'яти років спостерігається швидкий ріст кількості OTT-платформ, а світові карантинні заходи спричинили великий ріст навантаження на інтернет-інфраструктуру та протоколи передачі медіадних. Зростання обсягів потокового трафіку потребує нових підходів до управління мережевими ресурсами та покращення протоколів доставки медіа. Сучасні споживачі контенту не терплять збоїв в мережі, адже будь які проблеми відразу впливають на якість перегляду. Дані показують, що двосекундна затримка при буферизації через проблеми в мережах може призвести до втрати більше 20% аудиторії.

Одночасно з цим стає важче забезпечити безперебійну передачу мультимедійних потоків через різні мережеві протоколи. Сучасні мережі доставки контенту мають включати протоколи HTTP/2 та HTTP/3, впроваджувати адаптивні стрімінгові технології (HLS, MPEG-DASH), реалізовувати розумне розподіл навантаження та динамічно оптимізувати використання мережевої пропускної здатності для обслуговування мільйонів аудиторій.

Старі методи для спостереження за мережею і перевірки якості передачі медіа даних більше не відповідають новим вимогам до надійності та ефективності. Вони не можуть знайти складні мережеві проблеми, такі як нестабільність, втрата пакетів у певних частинах мережі, погіршення якості служби через перевантажені маршрутизери або труднощі із синхронізацією потоків на рівні транспортних протоколів.

Це пояснює потрібність створення розумних систем для спостереження в мережі, що з'єднують звичайні способи аналізу потоку з новими технологіями навчання машин. Такі рішення можуть не лише виявляти явні проблеми у мережах але й передбачати можливі місця заминок у інфраструктурі,

досліджувати якість медійного матеріалу з врахуванням особливостей мережевих правил.

Мета дослідження - це створити систему для моніторингу потоків OTT. Тут використовуються алгоритми штучного інтелекту щоб знайти та проаналізувати проблеми якоscті відео на рівні мережевих протоколів в реальному часі.

Досягнення встановленої цілі вимагає вирішення набору пов'язаних технічних завдань:

- провести аналіз сучасного стану інструментів для аналізу відеопотоків;
- розробити структурну схему проекту;
- розробити алгоритм роботи аналізу відеопотоків з використанням LLM мереж.
- провести порівняльний аналіз отриманих результатів.

Об'єктом дослідження є процеси спостереження мережі та контроль якості потоків відео по OTT в розподілених телекомунікаційних системах.

◌ Предметом дослідження є архітектурні принципи та програмні рішення для створення інтелектуальних систем моніторингу якості OTT-потоків з використанням гібридного підходу, що інтегрує комп'ютерний зір, аналіз мережевих метрик та обробку природної мови для автоматизації процесів технічного контролю медіаконтенту.

Технологічний набір для системи будується на нових інструментах для створення програм. Python вибрано як головну мову програмування через її добре розвинуту систему для даних та машинного навчання. OpenCV дає можливості для обробки відео потоків і реалізації алгоритмів комп'ютерного зору. NumPy має швидкі інструменти для матричних математичних обчислень та моделювання. FFmpeg використовується за декодуванні та аналізі мультимедійних потоків різноманітними форматами. Користувачський вигляд зроблено за допомогою фреймворка Tkinter, з шансом на розвиток до веб-техніки.

Практична цінність дослідження полягає в створенні готового до продажу програмного продукту який може сильно покращити обслуговування користувачів OTT-платформ і зменшити витрати провайдерів через автоматизацію процесів моніторингу мережі та аналізу якості медіа.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика мережевого моніторингу OTT-потоків

Контроль якість відео потоків в поширених OTT-системах є складним завданням для мережі. Це ускладнюється швидким зростанням трафіку і підвищеними вимогами до якості обслуговування. Головні проблеми можна поділити на кілька груп, кожна з яких потребує особливих рішень для мереж і оптимізацій протоколів.

Мережеві проблеми є важливою частиною труднощів з доставкою потокового контенту. Вони включають нестабільність TCP/UDP зв'язків, що призводить до розриву потоків або великого падіння якості передачі. Проблеми з маршрутизацією пакетів між вузлами CDN також грають важливу роль, особливо в часи великого трафіку під час великих трансляцій чи виходу популярного контенту. Аспекти кодування та стискання створюють певні виклики для мережевого переказу. Різні кодеки мають різну ефективність при передачі через різні мережеві протоколи, а налаштування, яке добре працює для одного типу медійного контенту може виявитися неприпустимім для іншого Наприклад, спортивні покази з рухомим змістом вимагають інших налаштувань протоколу, ніж статичні новинні програми.

Таблиця 1.1 - Класифікація проблем мережевого моніторингу OTT-потоків

Категорія проблем	Мережеві прояви	Вплив на QoE	Складність детекції
Протокольні збої	Втрата пакетів, тайм-аути TCP	Критичний	Низька
Проблеми кодування	Артефакти при декомпресії, блокування	Високий	Середня
Десинхронізація	Джиттер у потоках аудіо/відео	Високий	Середня

Деградація пропускної здатності	Зменшення битрейту, зниження роздільності	Середній	Висока
Зависання потоків	Блокування буферів, циклічні повтори	Критичний	Низька

Проблеми синхронізації звуку та відео в мережі дуже важливі для користувачів, бо людське сприйняття чутливе до різниць у часі. Навіть маленький джиттер у 40-50 мілісекунд може викликати ефект неприродності і погано вплинути на якість досвіду. Збільшення моніторингу в мережах створює особливі труднощі для архітектури систем, що розподілають данні. При обслуговуванні тисяч потоків традиційні способи контролю стають неефективними. Автоматичні системи мусять працювати з мінімальними затримками, обробляючи великі потоки даних і забезпечуючи швидку реакцію на знайдену проблему.

Різні типи пристроїв клієнтів роблять мережу ще більш складною. Один і той же потік може мати різну швидкість на нових мобільних машинах в порівнянні зі старими телевізорами. Система моніторингу мережі повинна враховувати ці особливості та перевіряти сумісність з різними пристроями.

Економічні сторони мережевих проблем дуже важливі. Кожна хвилина простою в мережі або поганої якості передачі може коштувати постачальнику великих грошей та шкоди репутації. Водночас, занадто дорогі системи моніторингу можуть бути нераціональні для малих і середніх операторів.

Правила в різних країнах також впливають на вимоги для спостереження. Деякі держави ставлять суворі потреби QoS для сервісів потокового відео, та постачальники повинні давати довідку про те що вони відповідають цим нормам. Людський фактор лишається важливим елементом навіть у дуже

автоматизованих мережевих система. Команди операторів мають швидко розуміти складні мережеві дані і робити добре рішення у критичних ситуаціях,.

Це потрібно не лише глибоких знань мережевих протоколів, а й досвіду роботи з певними типами медіа та обладнанням.

1.2 Характеристика предметної області

Область мережевого спостереження OTT-потоків включає широкий ряд мережевих технологій, протокольних рішень та методів, які забезпечують контроль якості доставки відео до кінцевих користувачів через IP-мережі. Розуміння цієї області вимагає аналізу всіх частин мережевої структури доставки контенту та їх протокольного співробітництва.

Архітектура сучасних OTT-систем побудована на принципах розподілу мережевої доставки вмісту. Центральні сервери зберігають оригінальний вміст у найкращій якості, що потім обробляється та підходить для різних умов мережі і типів пристроїв. Система доставки вмісту (CDN) забезпечує географічну диверсифікацію вмісту, покладаючи крайні сервери поруч з кінцевими користувачами щоб мінімізувати затримку мережі.

Транскодування - це процес зміни вихідного відео в кілька варіантів з різними якостями, швидкостями та кодеками, підготовленими для різних мережевих умов. Сучасні системи роблять від 5 до 15 простих профілів одного контенту, що дозволяє краще вибрати кращий варіант залежно від теперішньої пропускної здатності мережі користувача і здібностей його пристрою.

Таблиця 1.1 - Компоненти мережевої системи доставки OTT-контенту

Мережевий компонент	Протокольна функція	Критичні параметри	Методи мережевого моніторингу
Ingest-сервери	Прийом медіапотоків по RTMP/SRT	Пропускна здатність, джиттер	Аналіз TCP/UDP метрик, моніторинг буферів
Транскодери	Обробка через HTTP/HTTPS API	Латентність обробки, throughput	Порівняння вхідних/вихідних потоків, PSNR метрики
CDN edge-вузли	Кешування та HTTP/2 доставка	Доступність, RTT, cache hit ratio	HTTP health checks, мережевий pinging
ABR-контролери	Адаптивне переключення потоків	Точність bandwidth detection	Аналіз клієнтської телеметрії, QoE метрики
Клієнтські плеєри	HLS/DASH декодування	Стабільність буферизації, CPU usage	Клієнтські логи, performance API

Адаптивний біт-рейт стрімінг (ABR) є однією з найважчих частин системи для мережевого моніторингу. Алгоритми ABR завжди дивляться на умови мережі та можливості клієнта, швидко переключаючись між різними рівнями якості. Добрий мережевий моніторинг має слідкувати не лише за технічними метриками пропускної здатності, але і за якістю рішень алгоритмами ABR а також їх вплив на досвід користувача.

Клієнтські програми та медіаплеєри додають додатковий рівень складності до системи спостереження. Різні платформи мають відмінні можливості мережевих стеків, різні способи управління буферами та різні стратегії для обробки помилок в мережі. Система спостереження повинна враховувати особливості ці і збирати телеметричні дані з різних клієнтських пристроїв.

Мережева основа має важливу роль у якості доставки медіа. Інтернет-постачальники мають різні правила щодо обробки відеотрефіку, деякі можуть робити traffic shaping або ставити на перше місце інші види мережевих потоків. Якісний моніторинг мережі повинен відрізняти проблеми, які викликані внутрішньою мережею постачальника контенту неправильно, від труднощів ISP чи локальних умов для кінцевих користувачів

Аналіз і вимірювання якості мережевого досвіду стають все більш важливими для розуміння справжнього впливу мережевих проблем на людей. Звичайні мережеві оцінки, такі як використання пропускної здатності, або втрата пакетів, не завжди відповідають тому, як люди відчувають якість.

Об'єднання з робочими процесами робить систему мережевого нагляду частиною загальної IT-структури компанії. Мережеві дані для нагляду використовується не тільки для технічного контролю але й для прийняття стратегічних рішень про мережеві інвестиції, вибору провайдерів CDN та планування розвитку структури мережевого сервісу.

1.3 Аналіз існуючих рішень мережевого моніторингу OTT-потоків

Ринок рішень для моніторингу OTT-потоків показує широкий вибір товарів, від простих утиліт для перевірки мережевої доступності до складних платформ. Аналіз доступних рішень допомагає виявити основні тренди розвитку галузі та знайти добрі і погані сторони різних методів контролю якості відеопотоків. Сегмент бізнес-рішень має таких технологічних лідерів як Conviva, ThousandEyes, Catchpoint і SSIMWAVE? Ці платформи пропонують загальний підхід до моніторингу мережи який включає збір даних з мільйон клієнтських пристроїв, аналіз інфраструктури мережі а також надання докладної аналітики на якість досвіду користувачів.

Conviva має важливу роль на ринку завдяки своєму методу Реального Спостереження Користувачів (RUM), який збирає дані з мережі прямо з відеоплеєрів користувачів. Платформа досліджує більше ніж 500 мільярдів подій в мережі щороку та дає швидкі факти про якість роботи мережі. Головною перевагою цього методу є точність фактів про реальні умови в мережі у користувачів, але недолік - це залежність від інтеграції з програмами клієнтів та потреба зміни стеку мережевих плеєрів.

Рисунок 1.1 - Загальний огляд інтерфейсу Conviva

ThousandEyes зосереджується на всебічному мережевому моніторингу та користується розподіленою сіткою програмних агентів для перевірки роботи мережі і зв'язку з кількох географічних місць. Платформа дає змогу знайти проблеми у провайдерів Інтернету і мережах доставки контенту, але показує обмежені можливості для аналізу якості медіа. Їхній спосіб виявляється дуже ефективним для знаходження труднощів в доставці через складні мережеві схеми та багаторазову маршрутизацію.

Ssimwave показує новий метод спостереження, підкреслюючи сенсорне вивчення вмісту. Їх методи мають на меті імітувати, як люди оцінюють візуальну ясність та створити показники, які демонструють більш сильну відповідність особистим думкам людей щодо вмісту, який вони дивляться. Цей метод особливо корисний для тих, хто створює вміст і хоче покращити, наскільки добре вони використовують мережевий простір, зберігаючи високу якість своєї роботи.

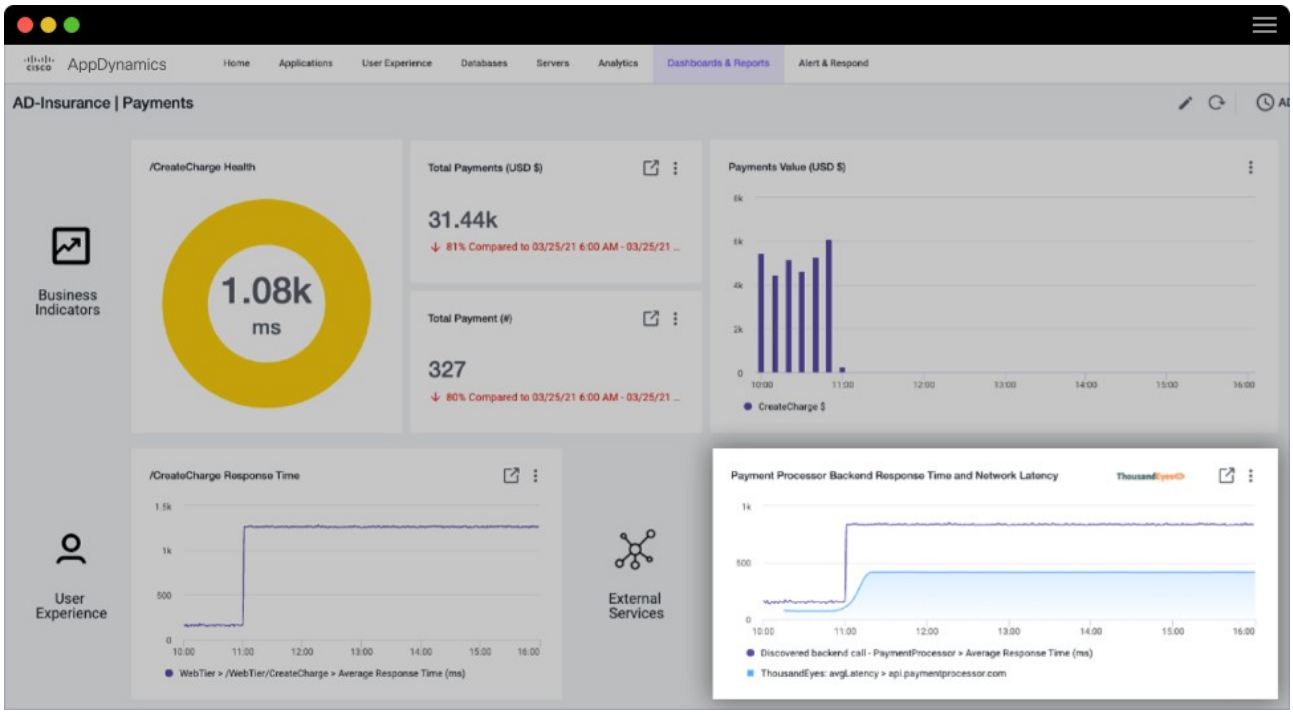

Рисунок 1.2 - Дашборд системи ThousandEyes

Група середнього класу показана інструментами, які зосереджуються на конкретних деталях моніторингу або обслуговують конкретні ринкові сфери. Ці предмети, як правило, коштують дешевше, але можуть запропонувати менше функцій, ніж системи бізнесу.

Telestream забезпечує конкретне обладнання для вивчення стандартів кодування відео та виявлення недоліків стиснення даних. Їх пункт Aurora пропонує ретельне вивчення відео послідовностей на етапі кадрів за кадром і

може помітити незначні проблеми якості, які можуть пропустити основні налаштування спостереження. Налаштування особливо добре працює для

перевірки якості у виробничих установах, де точність відтінку та дублювання компонентів має вирішальне значення.

Таблиця 1.2 - Аналіз спеціалізованих рішень середнього сегменту

Продукт	Спеціалізація	Переваги	Обмеження	Цільова аудиторія
Telestream Aurora	Аналіз якості кодування	Кадр-рівневий аналіз, детекція артефактів	Висока вартість ліцензій	Broadcasters, студії
Brightcove Analytics	Платформа-специфічна аналітика	Глибока інтеграція з Brightcove	Прив'язка до одної платформи	Клієнти Brightcove
JW Player Analytics	Фокус на веб-плеєри	Детальна аналітика поведінки	Обмежена підтримка live потоків	Веб-видавці
Wowza Streaming Engine	Серверний моніторинг	Реальний час, низька затримка	Потребує технічної експертизи	Стрімінгові провайдери

Brightcove Analytics підключається до відео -послуги під назвою Brightcove та пропонує розуміння того, скільки вмісту переглядається, якість відео та як користувачі взаємодіють з ним. Незважаючи на свої обмеження в середовищі Brightcove, система забезпечує глибоке розуміння користувачам цієї послуги та має очевидний макет для рекламодавців та наглядачів вмісту.

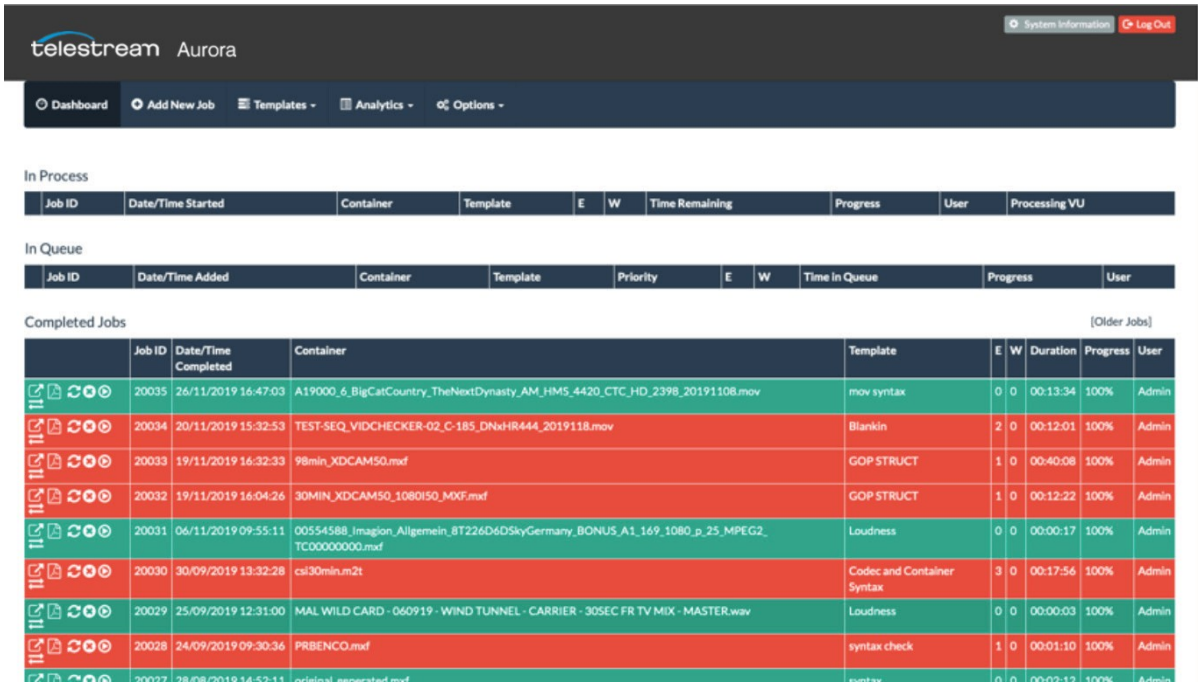

Рисунок 1.3 - Система Telestream Aurora.

Analytics JW Analytics зосереджується на даних Інтернет –відеоплеєра та пропонує всебічні уявлення про те, як глядачі взаємодіють з онлайн -матеріалом. Система спостерігає за статистикою участі в мережі, відхиляє проблеми з мережею та перевіряє ефективність розподілу реклами, що робить її особливо корисною для веб -видавців та рекламних платформ.

Частина з відкритим кодом займає велику частку ринку, особливо серед малих та середніх постачальників послуг, нових підприємств та шкіл. Ці інструменти забезпечують основну функціональність моніторингу мережі, але потребують значних технічних знань для коригування та налаштування.

FFMPEG продовжує залишатися основним інструментом для обробки потоків відео в домені програмного забезпечення з відкритим кодом. Його застосування для перевірки здоров'я мережі потребує створення власного коду та методів підключення, але пропонує повну команду за процедурою мережевого експертизи. Численні інструменти для бізнесу покладаються на FFMPEG для своїх розділів обробки даних в Інтернеті.

Таблиця 1.3 - Порівняння Open Source мережевих інструментів

Інструмент	Основна мережева функція	Переваги	Недоліки	Складність впровадження
FFmpeg	Multimedia network processing	Універсальність, безкоштовність	Потребує програмування	Висока
Nagios	Network infrastructure monitoring	Надійність, розширюваність	Застарілий інтерфейс	Середня
Zabbix	Comprehensive network monitoring	Потужна аналітика, активна спільнота	Обмежені відео-функції	Середня
Grafana + InfluxDB	Network metrics visualization	Красиві дашборди, гнучкість	Потребує додаткових компонентів	Середня

Nagios та Zabbix, хоча й не спеціалізуються на відеомоніторингу, широко використовуються для базового контролю доступності потоків та мережевої інфраструктури серверів. Ці системи мають розвинені можливості для network alerting та автоматизації, але потребують додаткових плагінів або розробки для роботи з відеоконтентом.

Рисунку 1.4 – Дашборд Grafana з метриками відеопотоків

Поєдання Grafana та InfluxDB тепер широко використовується для розробки персоналізованих налаштувань мережевого спостереження. Grafana пропонує сильні функції для вимірювання мережі, а InfluxDB ефективно зберігає ряд даних про продуктивність мережі на основі часу. Стек: стек мережевої панелі приладів.

Ця колекція дозволяє вам зібрати акуратно розроблені мережеві панелі мережі, але вимагає майстерності власних агентів збору даних для мережі. Останні три роки побачили активне доповнення AI Tech у інструментах відео -дорожнього годинника. Ця тенденція відкриває нові можливості для автоматизації мережевого аналізу та підвищення точності мережевих аномалій.

API Google Cloud Video Intelligence пропонує автоматичне обстеження вмісту, враховуючи умови передачі мережі, такі як сцена, об'єкт, виявлення тексту та оцінка якості після доставки. У сервісі використовуються міцні моделі машинного навчання, але встановлює обмеження на обробку в режимі реального часу і можуть бути дорогими з високими обсягами мережевого трафіку.

Amazon Video Rekognition забезпечує аналогічні функції фокусування для виявлення обличчя, аналізу емоцій та невідповідного вмісту, враховуючи

умови доставки мережі. Сервіс добре пов'язаний з середовищем AWS, але обмежувало здібності до ведення прямих трансляцій та потребує багато мережевої та обчислювальної потужності.

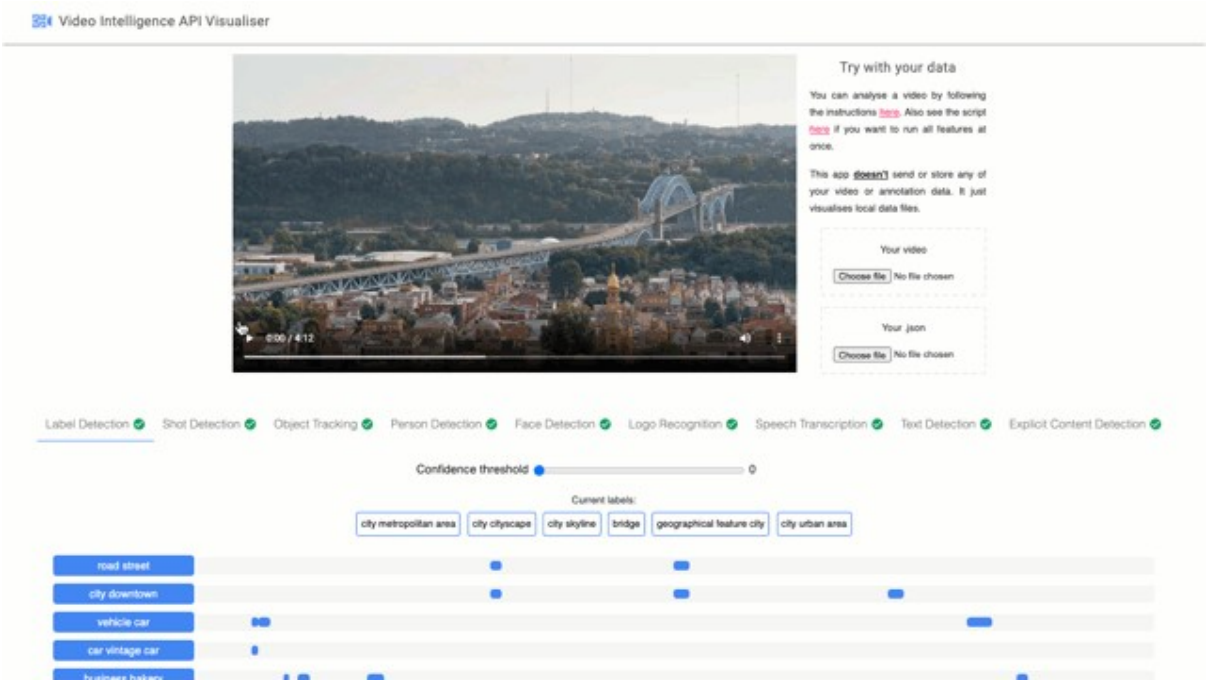

Рисунок 1.5 – Приклад роботи Google Video Intelligence API

MUX створив систему для аналізу потоку відео даних, зосереджуючись на легкому мережевому з'єднанні та розумінні розробника. Їх метод полягає у спрощенні складного відео- та мережевого аналізу для груп, які не мають досвіду у галузі відеотех та мережевих правил. Перший підхід API-це методологія, яка надає пріоритет розвитку API (інтерфейси програмування додатків) перед побудовою основної інфраструктури.

Qoe.ai представляє новий метод оцінки якості потокового потоку відео, використовуючи глибоке навчання, щоб імітувати, як люди сприймають ясність відео, враховуючи стан мережі. Їх методи аналізують великі набори даних за допомогою оцінки особистого судження та мають на меті прогнозувати, як користувачі реагуватимуть на різні зниження якості відео.

Місцеві стратегії мають вирішальне значення в регіонах з унікальними правилами або характеристиками мережі. Європейські фірми часто створюють методи з більшою увагою до конфіденційності та дотримання правил GDPR.

Корпоративні рішення часто включають значні витрати на професійні послуги мережевої інтеграції, які можуть становити 50-100% від вартості ліцензій. Ці витрати покривають інтеграцію з існуючими мережевими системами, навчання персоналу та ongoing технічну підтримку.

Головним трендом останніх років стає інтеграція машинного навчання для предиктивного мережевого аналізу та автоматичного виявлення network anomalies. Проте більшість існуючих рішень використовують ML лише для базової класифікації мережевих проблем, а не для глибокого аналізу візуального контенту в контексті мережевої доставки.

Edge computing набирає популярність як спосіб зменшення мережевих затримок та покращення якості аналізу. Розміщення аналітичних потужностей ближче до джерел контенту дозволяє швидше виявляти мережеві проблеми та зменшувати навантаження на центральні сервери.

Інтеграція з бізнес-аналітикою та customer experience платформами дозволяє поєднувати мережеві технічні метрики з бізнес-показниками та краще розуміти вплив мережевих технічних проблем на задоволеність клієнтів та фінансові результати.

2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Розробка мережевої архітектури програмної системи

Для створення системи мережевого моніторингу OTT-потоків обрано розподілену мікросервісну архітектуру, яка забезпечує необхідну мережеву гнучкість та горизонтальну масштабованість. Ця архітектурна модель дозволяє розділити мережевий функціонал на автономні компоненти, кожен з яких відповідає за специфічні завдання аналізу та моніторингу відеопотоків через мережеві протоколи. Основними принципами мережевої архітектури є модульність для незалежної розробки network-компонентів, слабка мережева зв'язаність через стандартизовані REST API та gRPC протоколи, висока мережева доступність через реплікацію критичних сервісів та динамічна масштабованість для обробки піків мережевого трафіку.

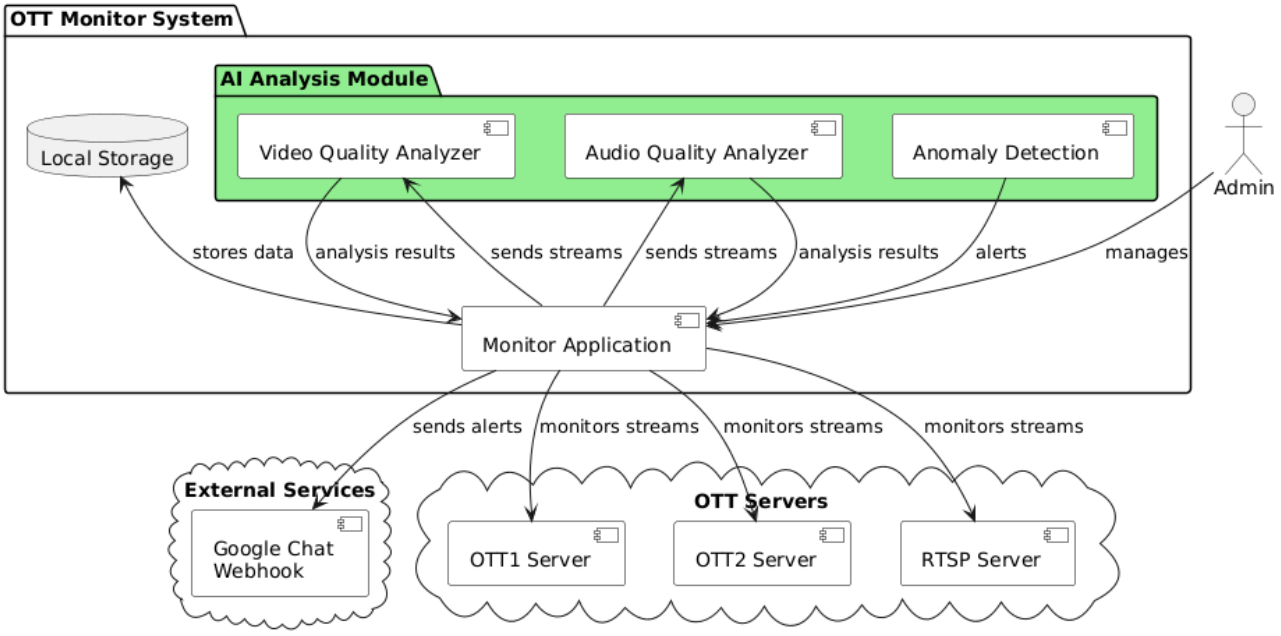

Рисунок 2.1 – Загальна мережева архітектура системи моніторингу OTT-потоків

Мережева архітектура включає presentation layer з веб-інтерфейсом та API gateway, business logic layer з основними мережевими мікросервісами, data layer з розподіленими базами даних та infrastructure layer з container orchestration.

Система складається з спеціалізованих мережевих мікросервісів. Stream Management Service відповідає за конфігурацію джерел відео та налаштування параметрів мережевого моніторингу. Network Monitoring Service здійснює перевірку мережевої доступності та збір технічних метрик через TCP/UDP протоколи.

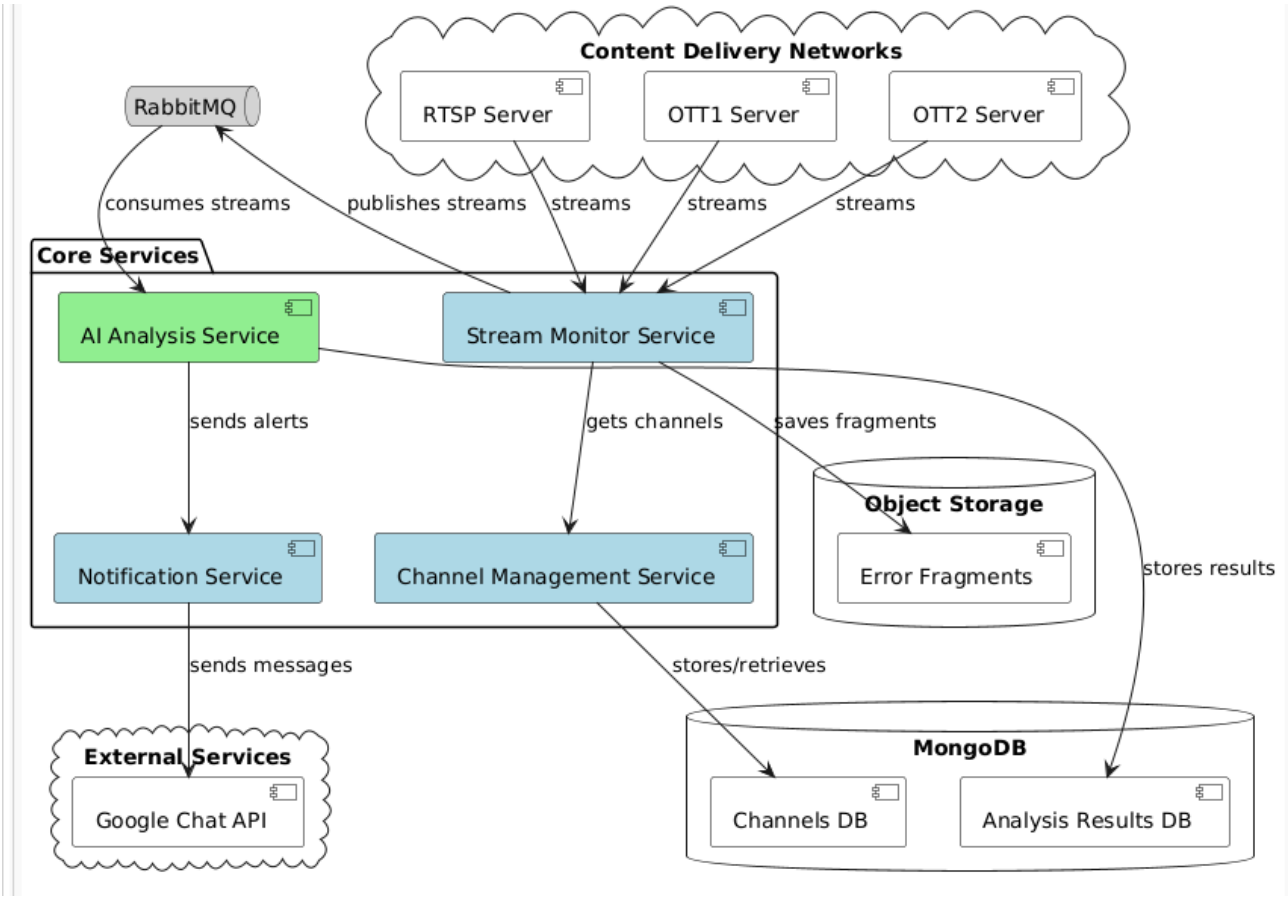

Рисунок 2.2 – Архітектура мережевих мікросервісів

AI Analysis Service використовує алгоритми комп'ютерного зору для аналізу візуального контенту з урахуванням мережевих умов доставки. Notification Service забезпечує доставку повідомлень через різні мережеві канали комунікації (HTTP webhooks, SMTP, WebSockets). Система використовує гібридний підхід до мережевого зберігання даних. PostgreSQL зберігає конфігураційні дані мережевих підключень та метадані користувачів. InfluxDB оптимізована для часових рядів мережевих метрик моніторингу (bandwidth, latency, packet loss). Redis використовується для мережевого кешування та

управління сесіями користувачів.

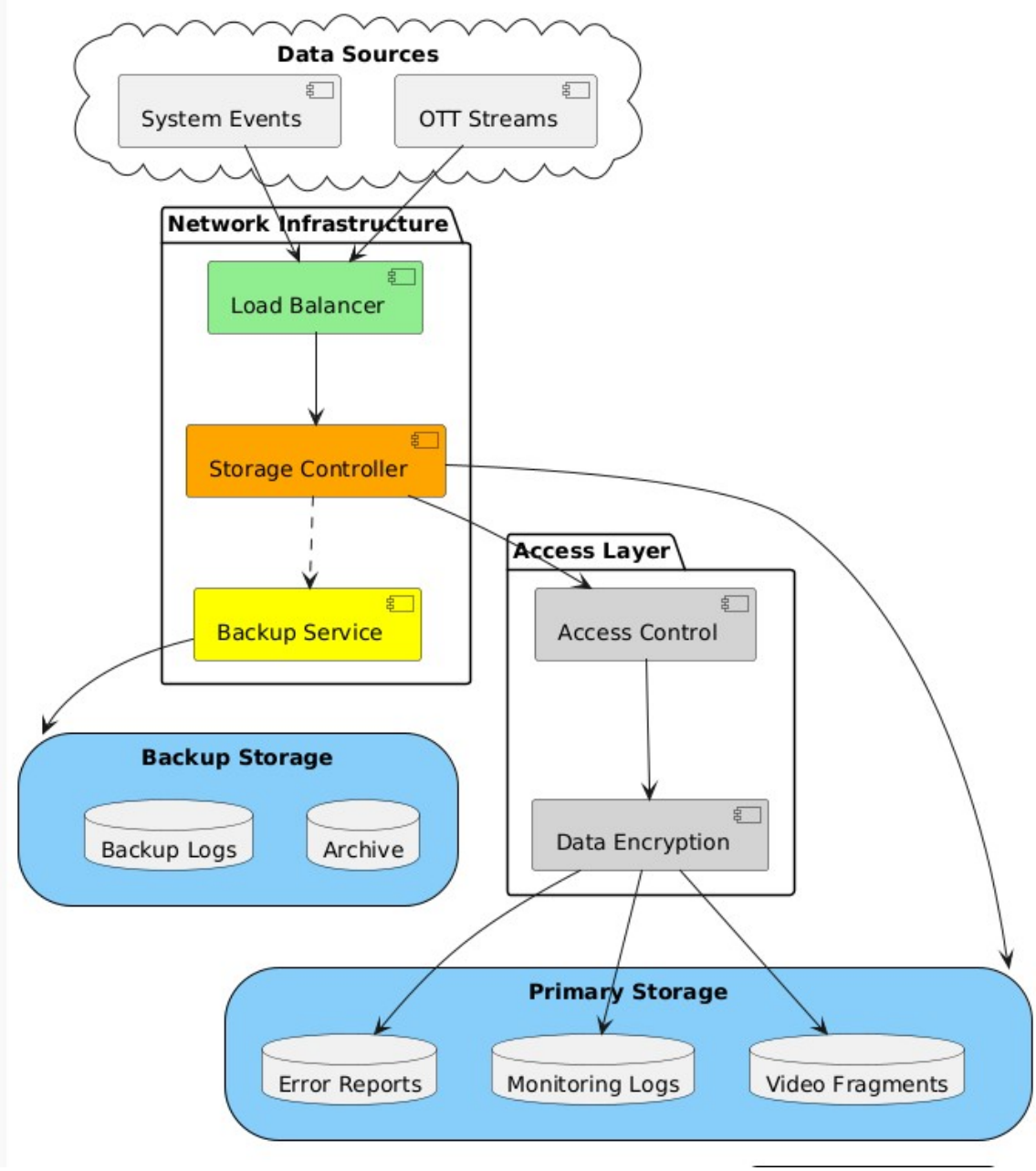

Рисунок 2.3 – Схема мережевого зберігання даних

MongoDB зберігає результати ШІ-аналізу з неструктурованими даними про мережеву якість. Object Storage містить мультимедійні дані для мережевого аналізу та архівування.

API Gateway забезпечує єдину точку мережевого входу з автентифікацією, маршрутизацією запитів та rate limiting. Система підтримує RESTful API через HTTPS, WebSocket з'єднання для real-time updates та GraphQL інтерфейс для гнучких запитів. Мережеві інтеграційні можливості включають webhook

інтерфейси для зовнішніх систем та SNMP протокол для взаємодії з мережевим обладнанням.

Архітектура мережевої безпеки базується на принципі zero trust з множинними методами автентифікації. Рольова модель контролю доступу забезпечує гнучке налаштування мережевих прав користувачів.

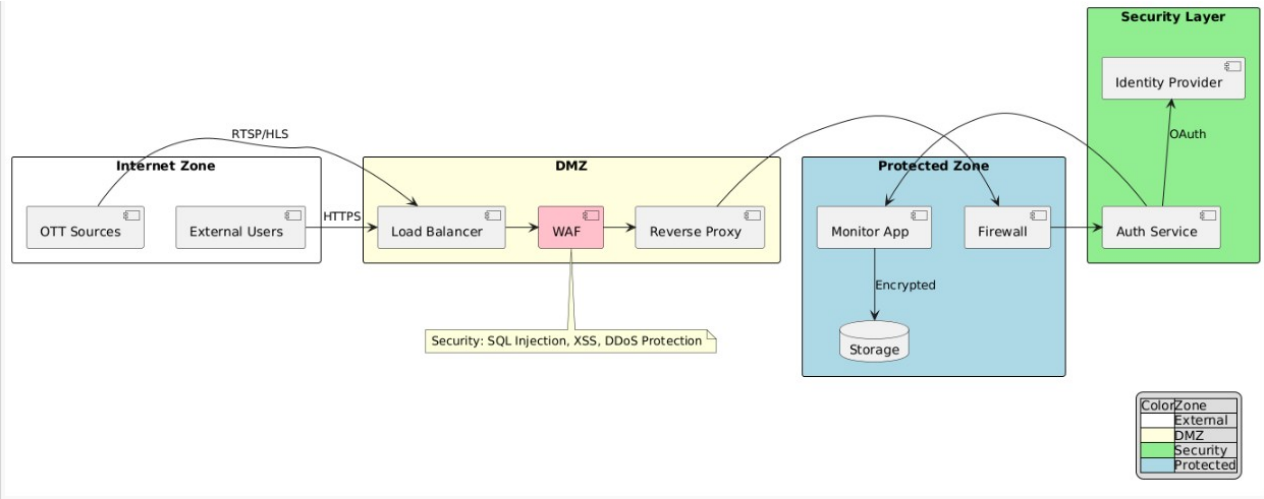

Рисунок 2.4 – Схема мережевої безпеки та контролю доступу

Система включає TLS/SSL шифрування мережевого трафіку, автоматичну ротацію API ключів та аудит мережевих операцій доступу.

2.2 Проектування моделі мережевого аналізу

Модель мережевого аналізу структурована у вигляді ієрархії з чотирма рівнями обробки. Перший рівень включає базові мережеві перевірки доступності потоку через HTTP/HTTPS протоколи та валідацію мережевих метаданих. Другий рівень аналізує технічні параметри мережевого потоку (throughput, jitter, RTT).

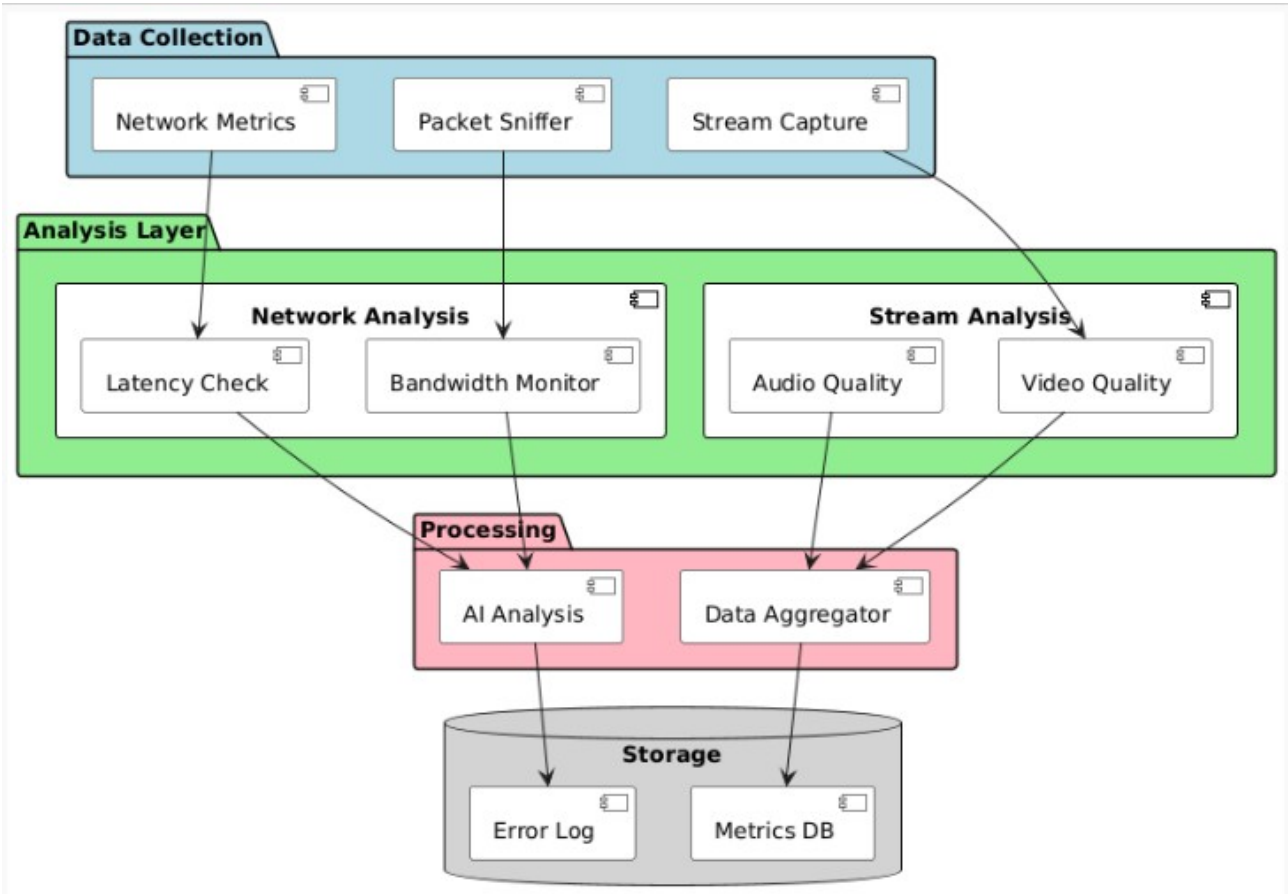

Рисунок 2.5 – Багаторівнева модель мережевого аналізу

Третій рівень застосовує комп'ютерний зір для аналізу візуального контенту з урахуванням мережевих умов передачі. Четвертий рівень використовує машинне навчання для інтелектуального аналізу контенту та прогнозування мережевих проблем.

Система технічного мережевого моніторингу комбінує детерміністичні алгоритми та статистичні методи. Виявлення мережевих перерв потоку використовує HTTP health checking з адаптивними інтервалами та exponential backoff.

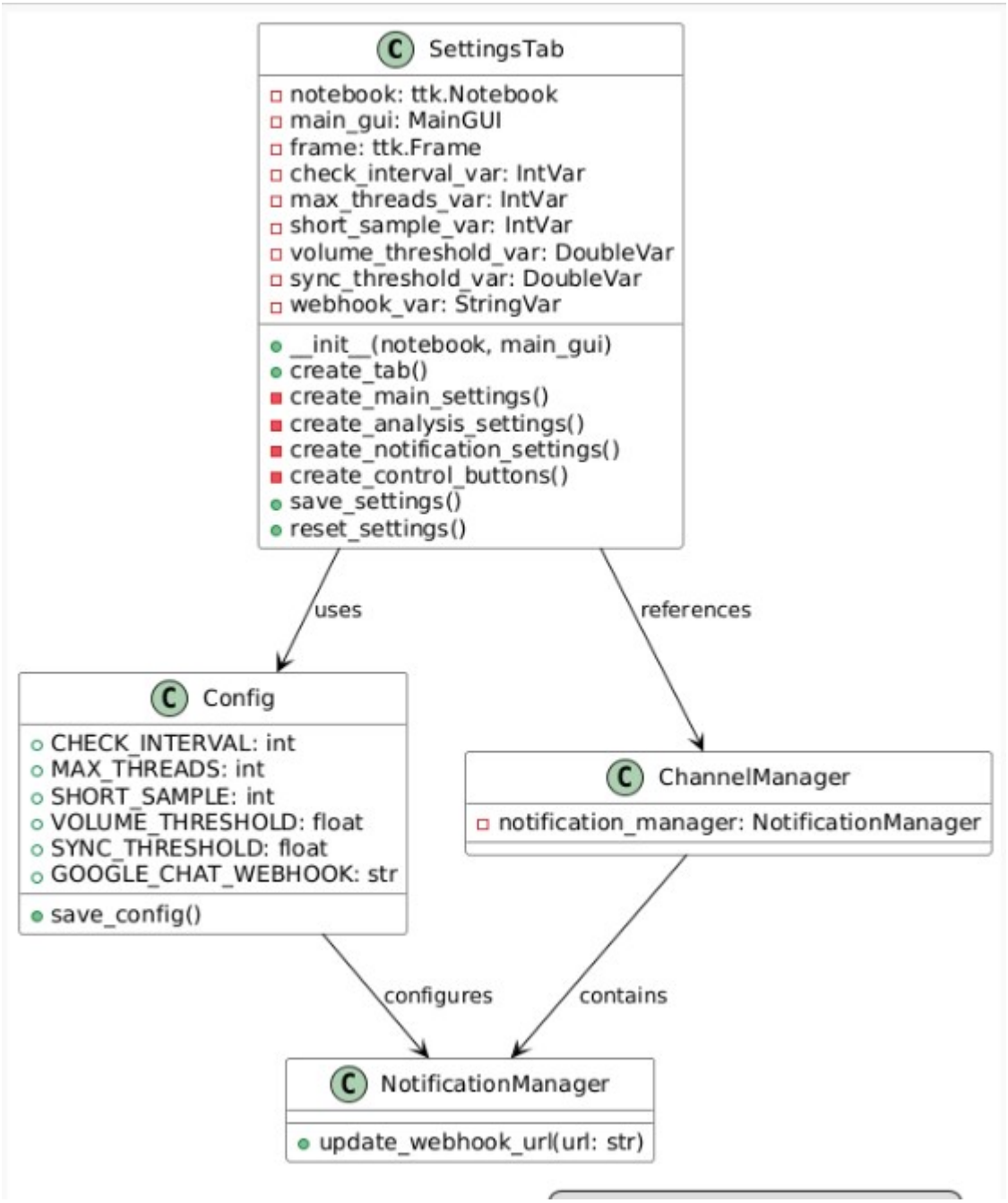

Рисунок 2.6 – Виявлення мережевих аномалій бітрейту

Аналіз стабільності мережевого битрейту використовує sliding window algorithms для виявлення bandwidth fluctuations. Контроль мережевої синхронізації аудіо-відео аналізує timestamp drift у потоці та виявляє network-induced desync.

Ядро візуального аналізу базується на згорткових нейронних мережах, адаптованих для відеоконтенту з урахуванням мережевих артефактів. Архітектура включає просторові та темпоральні модулі для аналізу network-

induced degradations.

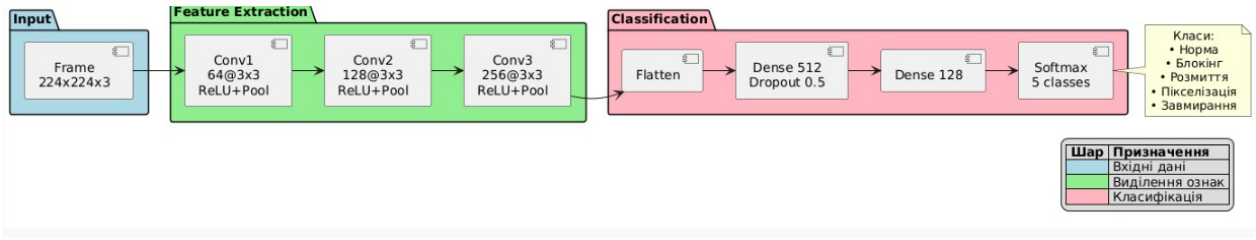

Рисунок 2.7 – Архітектура CNN для детекції мережевих артефактів

Детекція мережевих артефактів стиснення використовує навчену модель для розпізнавання JPEG та H.264/H.265 compression artifacts, спричинених network congestion. Аналіз різкості комбінує традиційні фільтри з глибоким навчанням для виявлення blur effects від packet loss.

Інтелектуальна класифікація дозволяє адаптувати пороги виявлення залежно від типу контенту та мережевих умов. Модель побудована на архітектурі ResNet з темпоральними модулями для аналізу network-sensitive content.

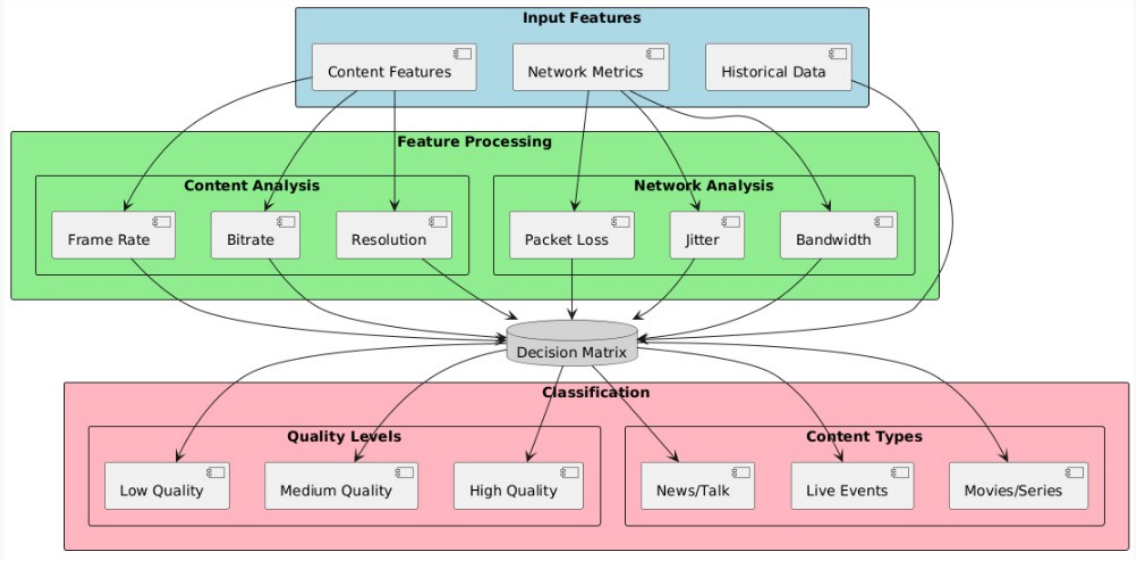

Рисунок 2.8 – Модель класифікації типів контенту з урахуванням мережевих факторів

Система розрізняє новини, спорт, фільми та інші типи контенту, автоматично підлаштовуючи чутливість детекторів для кожного типу з урахуванням мережевих особливостей доставки.

2.3 Алгоритмізація та логіка мережевого аналізу

Процес мережевого аналізу організований як distributed processing pipeline з паралельною архітектурою. Початковий етап включає мережеве декодування сегментів та попередню обробку кадрів з урахуванням network latency.

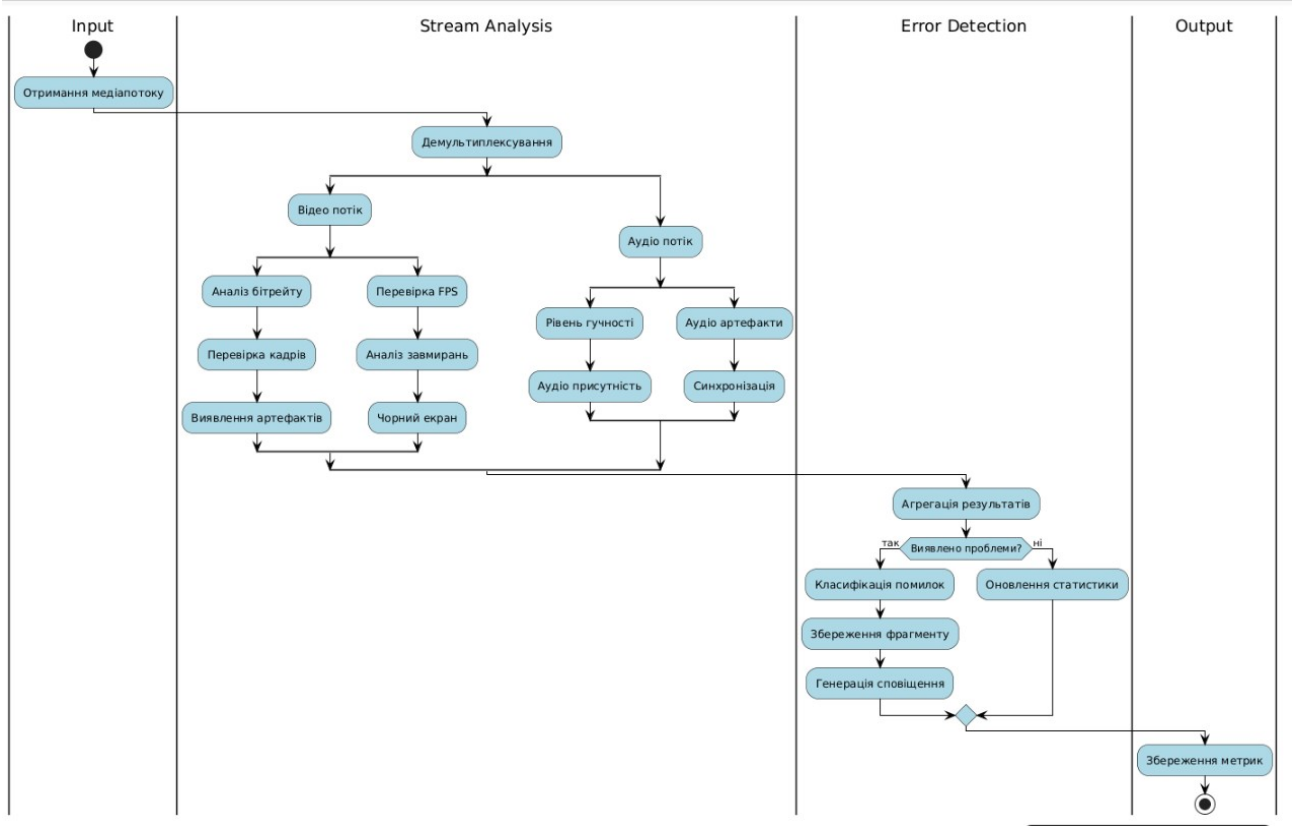

Рисунок 2.9 – Загальна схема алгоритму мережевого аналізу

Адаптивне семплування оптимізує мережеві ресурси через інтелектуальний вибір кадрів базуючись на network conditions. Система пріоритизації забезпечує розподіл мережевих ресурсів між потоками залежно від

їх критичності.

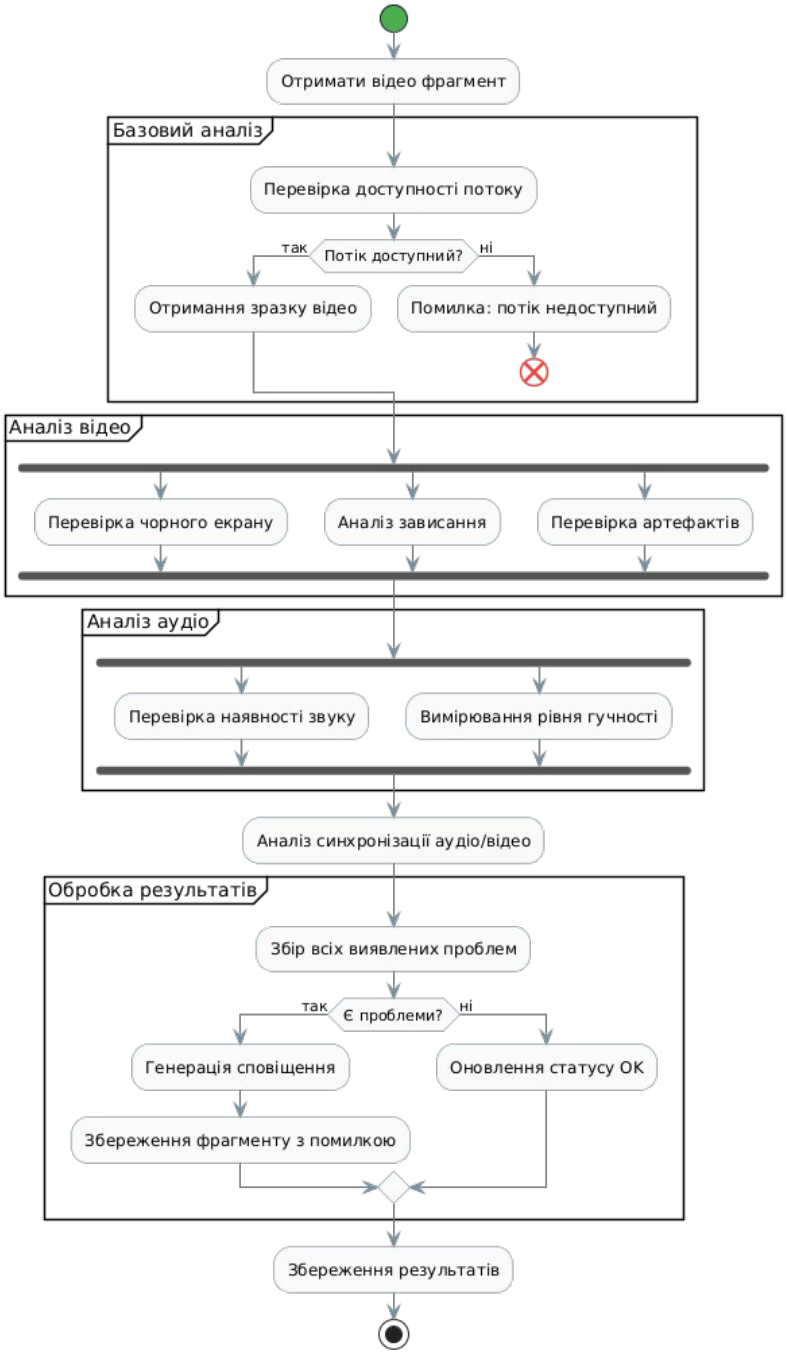

Рисунок 2.10 – Алгоритм обробки відеопотоку

Етап попередньої обробки включає нормалізацію відеоданих для консистентності мережевого аналізу. Система автоматично детектує параметри мережевого потоку та застосовує відповідні перетворення для компенсації network-induced distortions.

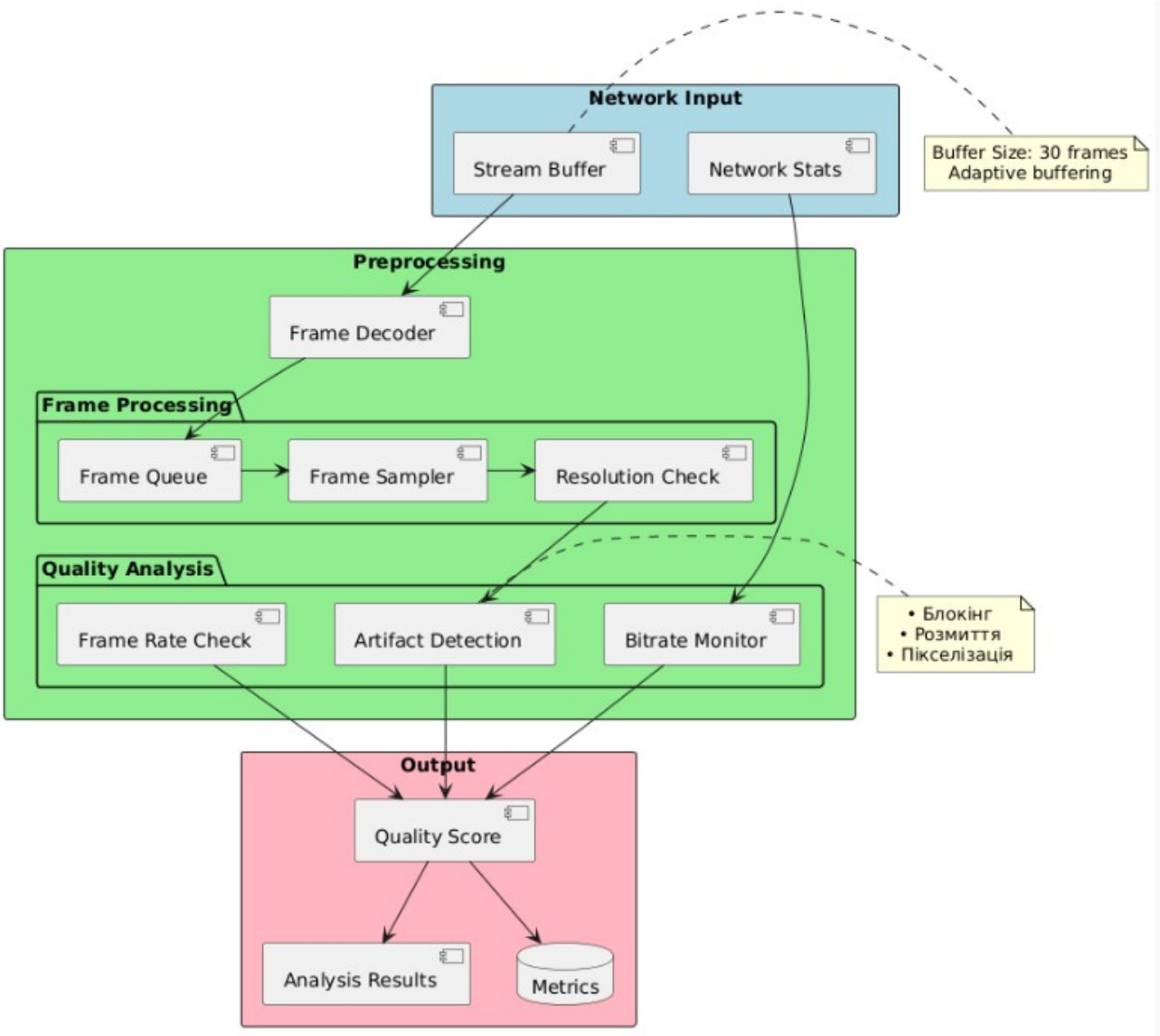

Рисунок 2.11 – Схема попередньої обробки відеоданих з урахуванням мережевих факторів

Масштабування використовує високоякісні алгоритми інтерполяції з урахуванням мережевих обмежень. Корекція колірного простору забезпечує уніфікацію для алгоритмів аналізу та компенсує network-induced color shifts.

Детекція мережевих аномалій базується на комбінації традиційних методів та глибокого навчання. Початковий рівень включає швидкі статистичні перевірки на мережеві збої, такі як чорні кадри від connection drops та compression artifacts від bandwidth limitations.

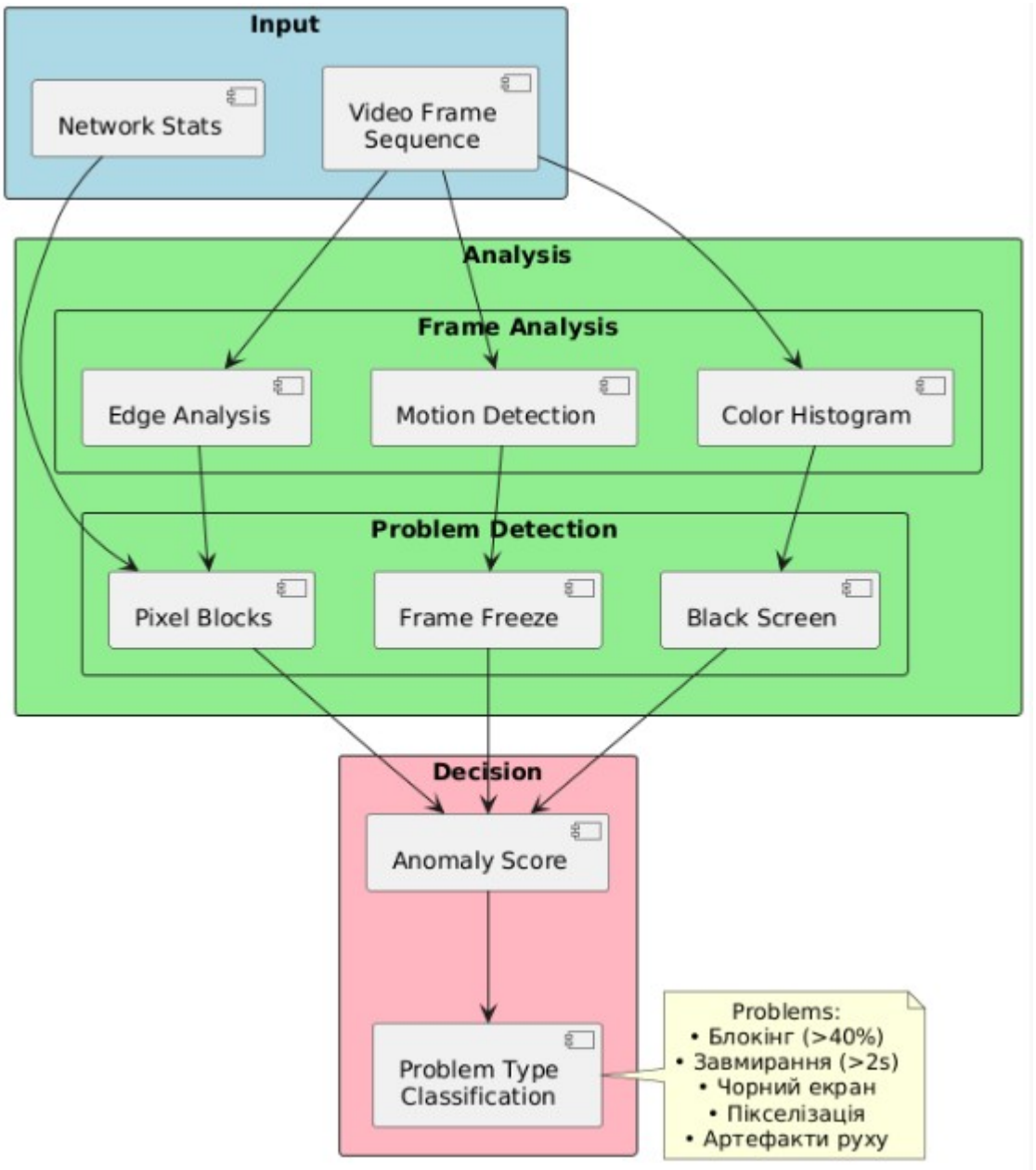

Рисунок 2.12 – Алгоритм детекції візуальних аномалій, спричинених мережевими проблемами

Аналіз градієнтів виявляє проблеми з різкістю через packet loss. Детекція блокових артефактів використовує частотний аналіз для виявлення compression artifacts від network congestion. Алгоритм застиглих кадрів порівнює послідовні кадри та виявляє buffer underruns.

Алгоритм мережевої пріоритизації враховує критичність каналу, тип мережевої проблеми та поточну аудиторію. Матриця критичності визначає важливість різних мережевих проблем базуючись на їх впливі на QoE.

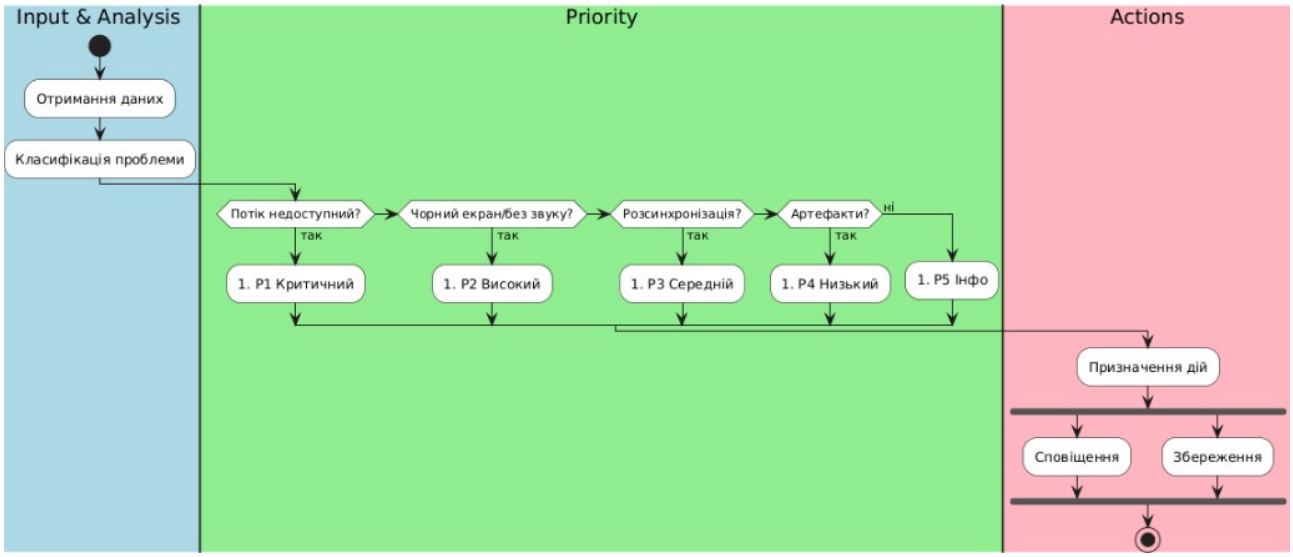

Рисунок 2.13 – Алгоритм пріоритизації мережевих проблем

Автоматична ескалація забезечує своєчасне залучення мережевих інженерів. Інтелектуальне групування об'єднує пов'язані мережеві проблеми для запобігання спаму сповіщень та коректної ідентифікації root cause.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Програмна реалізація проєкту

Програмна реалізація системи мережевого моніторингу OTT-потоків представляє собою комплексне рішення, що поєднує традиційні методи технічного аналізу медіа-потоків з інноваційними можливостями штучного інтелекту. Система побудована за модульним принципом з використанням мови програмування Python та спеціалізованих бібліотек для обробки мережевого медіа-контенту, що забезпечує гнучкість архітектури та можливість масштабування.

Основу системи складає мова програмування Python версії 3.8 або вище, яка була обрана завдяки своїй універсальності та наявності потужних бібліотек для роботи з мережевими медіа-потоками. Python забезпечує відмінну підтримку асинхронного програмування через модуль asyncio, що є критично важливим для одночасного мережевого моніторингу множини потоків. Екосистема Python також надає розвинену підтримку для роботи з TCP/UDP протоколами та HTTP/HTTPS обробки відео, що дозволяє системі ефективно взаємодіяти з різноманітними джерелами потокового контенту.

Мережева архітектура системи спроектована таким чином, щоб забезпечити максимальну гнучкість мережевих з'єднань, horizontal scalability та простоту подальшого розвитку network-функціоналу. Це досягається через використання асинхронних патернів програмування, що дозволяють системі одночасно обробляти сотні потоків без значного навантаження на системні ресурси.

Для створення мережевого графічного інтерфейсу користувача використовується бібліотека Tkinter з додатковими network-aware компонентами. Tkinter надає всі необхідні компоненти для створення повнофункціонального network monitoring додатку з інтуїтивно зрозумілим інтерфейсом, а модуль ttk додатково використовується для створення сучасного

вигляду інтерфейсу з network status indicators.

Центральним компонентом системи є FFmpeg як потужний інструмент для роботи з мережевими мультимедійними файлами та потоками. FFmpeg використовується як зовнішня утиліта для network stream capture, їх мережевого аналізу та виявлення network-induced quality issues. Вибір FFmpeg обумовлений його універсальністю та підтримкою всіх сучасних мережевих протоколів потокового відео, включаючи RTSP over TCP/UDP, HLS over HTTP/HTTPS, RTMP та adaptive streaming protocols.

Файл main.py виконує роль точки входу в додаток та реалізує комплексну систему перевірки мережевих залежностей. Цей модуль відповідає за ініціалізацію всіх необхідних компонентів системи та забезпечує правильну послідовність запуску різних підсистем. Особливу увагу приділено перевірці доступності мережевих утиліт та бібліотек, що є критично важливими для функціонування системи моніторингу.

```
def check_network_dependencies():
    """Перевіряє наявність необхідних мережевих залежностей"""
    required_modules = ['socket', 'urllib', 'requests', 'asyncio', 'concurrent.futures']
    missing_modules = []
      for module in required_modules:
       try:
           __import__(module)
       except ImportError:
           missing_modules.append(module)
    network_tools = ['ping', 'traceroute']
```

Модуль network_config.py реалізує централізовану систему управління мережевою конфігурацією, яка автоматично адаптується до різних типів мережевих потоків та забезпечує гнучке налаштування параметрів network monitoring. Цей компонент містить логіку розпізнавання типів потоків та

відповідного налаштування параметрів FFmpeg для оптимальної роботи з кожним протоколом.

Система автоматично визначає тип потоку за URL та застосовує відповідні оптимізації. For RTSP потоків налаштовуються специфічні параметри транспорту, тайм-аутів та буферизації, тоді як для HLS потоків увага приділяється налаштуванню протокольних whitelist'ів та механізмів автоматичного перепідключення при мережевих проблемах.

```
    stream_type = self.detect_network_stream_type(stream_url)
    base_cmd = ["ffmpeg", "-y"]
        if stream_type == 'rtsp':
        base_cmd.extend(["-rtsp_transport", "tcp", "-timeout", "10000",
                         "-buffer_size", "1024000", "-max_delay", "500000"])
    elif stream_type == 'hls':
        base_cmd.extend(["-protocol_whitelist", "file,http,https,tcp,tls",
                         "-reconnect", "1", "-reconnect_streamed", "1"])
        return base_cmd + ["-i", stream_url, "-t", str(duration), "-c", "copy"]
```

Основний клас NetworkStreamChecker відповідає за аналіз мережевої якості потоків та виявлення різноманітних network-related проблем. Цей компонент реалізує складну логіку перевірки network connectivity, bandwidth analysis, latency measurement та інших мережевих параметрів потоку. Особливе значення має функція аналізу аудіо-якості, яка враховує вплив мережевих проблем на цілісність аудіо-потоку.

Найбільш інноваційним аспектом системи є інтеграція великої мовної моделі Claude від Anthropic, що представляє собою принципово новий підхід до автоматизації аналізу та прийняття рішень в процесі моніторингу медіа-потоків. Ця інтеграція дозволяє системі не просто виявляти технічні проблеми, але й надавати глибокий контекстуальний аналіз причин їх виникнення та рекомендації щодо усунення.

LLM-компонент побудований навколо спеціалізованого класу

NetworkLLMAnalyzer, який інтегрується з основною системою мережевого моніторингу та забезпечує інтелектуальну обробку результатів перевірки потоків

з урахуванням мережевих метрик. Система використовує контекстний мережевий аналіз для накопичення знань про типові мережеві проблеми та їх вирішення, створюючи персоналізовані рекомендації на основі історичних даних конкретної системи.

```
def __init__(self, api_key):
    self.client = anthropic.Anthropic(api_key=api_key)
    self.network_conversation_history = []
    self.network_problem_patterns = {}
    async def analyze_network_stream_issues(self, channel_data, network_metrics, historical_data)
```

Особливо цінною є функція автоматичного створення мережевої документації та інструкцій на основі виявлених мережевих проблем. LLM аналізує типові мережеві сценарії помилок та генерує покрокові інструкції для їх усунення, що значно підвищує ефективність роботи технічних спеціалістів. Система здатна створювати детальні керівництва, які включають діагностичні команди, послідовність дій для усунення проблем та профілактичні заходи.

Інтеграція LLM з системою мережевих сповіщень дозволяє створювати більш інформативні та контекстуальні повідомлення про мережеві проблеми. Замість стандартних технічних алертів, система генерує зрозумілі повідомлення, які містять не лише опис проблеми, але й аналіз можливих причин, рекомендовані дії та прогноз впливу на користувачів.

```
async def generate_network_maintenance_report(self, period_data, network_infrastructure):
    """Генерує звіт про мережеве технічне обслуговування"""
    report_prompt = f"""
    Створи детальний звіт про стан мережевої інфраструктури:
```

Статистика: {period_data['network_stats']}

Критичні інциденти: {period_data['network_critical_issues']}

Топологія мережі: {network_infrastructure}"""

Система також реалізує предиктивний мережевий аналіз, де LLM використовує історичні мережеві дані для прогнозування потенційних мережевих проблем. Це дозволяє переходити від реактивного до проактивного підходу в управлінні мережевою інфраструктурою, попереджуючи проблеми до їх виникнення.Клас NetworkChannelManager координує весь процес мережевого моніторингу та управляє network state системи. Він реалізує асинхронну архітектуру для одночасної перевірки декількох каналів та забезпечує розумну логіку network-aware сповіщень. Цей компонент є серцем системи, яке синхронізує роботу всіх підсистем та забезпечує коректну обробку результатів моніторингу.

Графічний інтерфейс реалізовано через модульну систему вкладок з network-aware компонентами, де кожна вкладка відповідає за окремий аспект мережевої функціональності. Головний клас NetworkOTTMonitorGUI координує роботу всіх компонентів мережевого інтерфейсу, забезпечуючи зручну та інтуїтивну взаємодію користувача з системою.

Система мережевих сповіщень реалізована через клас NetworkNotificationManager, який інтегрується з Google Chat через HTTPS webhooks для надсилання повідомлень про виявлені мережеві проблеми. Система розумно групує повідомлення та уникає спаму з урахуванням мережевого стану, використовуючи алгоритми для визначення критичності проблем та оптимальної частоти сповіщень.

Загалом, програмна реалізація представляє собою високотехнологічне рішення, яке поєднує надійність традиційних методів моніторингу з інноваційними можливостями штучного інтелекту, створюючи унікальну систему для професійного моніторингу OTT-потоків з unprecedented рівнем

автоматизації та інтелектуального аналізу.

3.2 Тестування роботи модуля

Тестування системи моніторингу OTT потоків представляє собою комплексний процес перевірки функціональності, надійності та ефективності всіх компонентів системи в різних умовах експлуатації. Враховуючи критичність завдань, які вирішує система, було розроблено багаторівневу стратегію тестування, що включає модульне, інтеграційне, функціональне та навантажувальне тестування.

Модульне тестування було проведено для кожного окремого компоненту системи з метою перевірки коректності роботи індивідуальних функцій та методів. Особлива увага приділялася тестуванню критичних модулів, таких як аналізатор потоків, менеджер каналів та система сповіщень. Для цього було створено спеціалізований набір тестових сценаріїв, що покривають різні аспекти функціональності кожного модуля.

Тестування модуля StreamChecker включало перевірку правильності визначення типу потоку для різних URL адрес. Було створено тестовий набір, що включав RTSP, HLS, RTMP та HTTP потоки:

```
def test_stream_type_detection():
    """Тестує правильність визначення типу потоку"""
    test_cases = [
        ("rtsp://192.168.1.100:554/stream", "rtsp"),
        ("https://example.com/playlist.m3u8", "hls"),
        ("rtmp://live.example.com/stream", "rtmp"),
        ("http://example.com/stream.mp4", "http")
```

```
]
```

Тестування аналізу аудіо компонентів проводилося з використанням спеціально підготованих тестових файлів з різними рівнями гучності. Це дозволило перевірити правильність роботи алгоритму виявлення відсутності звуку:

```
def test_audio_analysis():
    """Тестує аналіз аудіо компонентів"""
    # Тестовий файл без звуку
    silent_file = "test_files/silent_video.mp4"
    error_flag = {"audio": False}

    result = analyzer.check_audio(silent_file, "Test Channel",
                                  error_flag, "test")
```

Для тестування системи управління конфігурацією було створено окремий набір тестів, що перевіряє правильність завантаження, збереження та валідації налаштувань:

```
def test_config_management():
    """Тестує управління конфігурацією"""
    # Створюємо тестову конфігурацію
    test_config = {
        "SERVERS": {
            "test_server": {
                "name": "Test Server",
                "base_url": "https://test.example.com",
            }
        }
    }
```

Інтеграційне тестування було зосереджено на перевірці взаємодії між

різними компонентами системи. Особливо важливим було тестування інтеграції між менеджером каналів та системою сповіщень, оскільки від їх злагодженої роботи залежить своєчасне інформування про проблеми.

Тестування системи сповіщень проводилося з використанням тестового webhook Google Chat, що дозволило перевірити правильність формування та відправки повідомлень без впливу на робочі канали:

```
def test_notification_system():
    """Тестує систему сповіщень"""
    notification_manager = NotificationManager()
    notification_manager.webhook_url = "https://test-webhook-url"

    # Тест простого повідомлення
    test_message = "Тестове повідомлення про проблему каналу Test Channel"
    thread_key = notification_manager.send_to_google_chat(test_message)

    assert thread_key is not None, "Повідомлення не відправлено"
    assert thread_key.startswith("thread_"), "Неправильний формат thread_key"
```

Функціональне тестування охоплювало перевірку всіх основних сценаріїв використання системи. Було розроблено детальні тест-кейси для кожної функціональної можливості системи, включаючи додавання каналів, запуск моніторингу, обробку помилок та генерацію звітів. Навантажувальне тестування було проведено для визначення максимальної кількості каналів, які система може одночасно моніторити без втрати якості аналізу. Тестування проводилося поетапно з поступовим збільшенням кількості каналів від 10 до 100.

```
    def simulate_channel_check(channel_id):
        """Симулює перевірку каналу"""
        start_time = time.time()
        # Симуляція роботи StreamChecker
        time.sleep(random.uniform(5, 15))  # Час перевірки каналу
```

```
    end_time = time.time()
    return channel_id, end_time - start_time
```

Результати навантажувального тестування показали, що система ефективно обробляє до 50 каналів одночасно при налаштуванні MAX_THREADS = 3. При збільшенні кількості каналів до 100 спостерігається зростання часу обробки, але система залишається стабільною.

Тестування системи попереднього перегляду відео проводилося з різними типами потоків для перевірки сумісності з різними протоколами та кодеками:

```
def test_video_preview():
    """Тестує систему попереднього перегляду"""
    test_streams = [
        "rtsp://test.example.com:554/stream",
        "https://test.example.com/playlist.m3u8",
        "http://test.example.com/stream.mp4"
    ]
```

3.3 Огляд графічної інтерфейсу

Робота з системою моніторингу OTT потоків починається з запуску головного додатку через файл main.py або відповідний ярлик на робочому столі. При першому запуску система автоматично перевіряє наявність всіх необхідних компонентів та створює базову конфігурацію. Користувач побачить головне вікно додатку з набором вкладок, кожна з яких відповідає за певний аспект роботи системи. Початкове налаштування системи розпочинається з конфігурації серверів. Для цього необхідно перейти на вкладку "Сервери" та додати інформацію про OTT сервери, які потрібно моніторити. У формі додавання сервера вказується унікальний ідентифікатор сервера, його назва для

відображення в інтерфейсі, базовий URL сервера та статус увімкнення. Наприклад, для додавання першого OTT сервера вводимо ID "ott1", назву "Основний OTT сервер", URL "https://ott1.columbus.te.ua" та відмічаємо прапорець "Увімкнено".

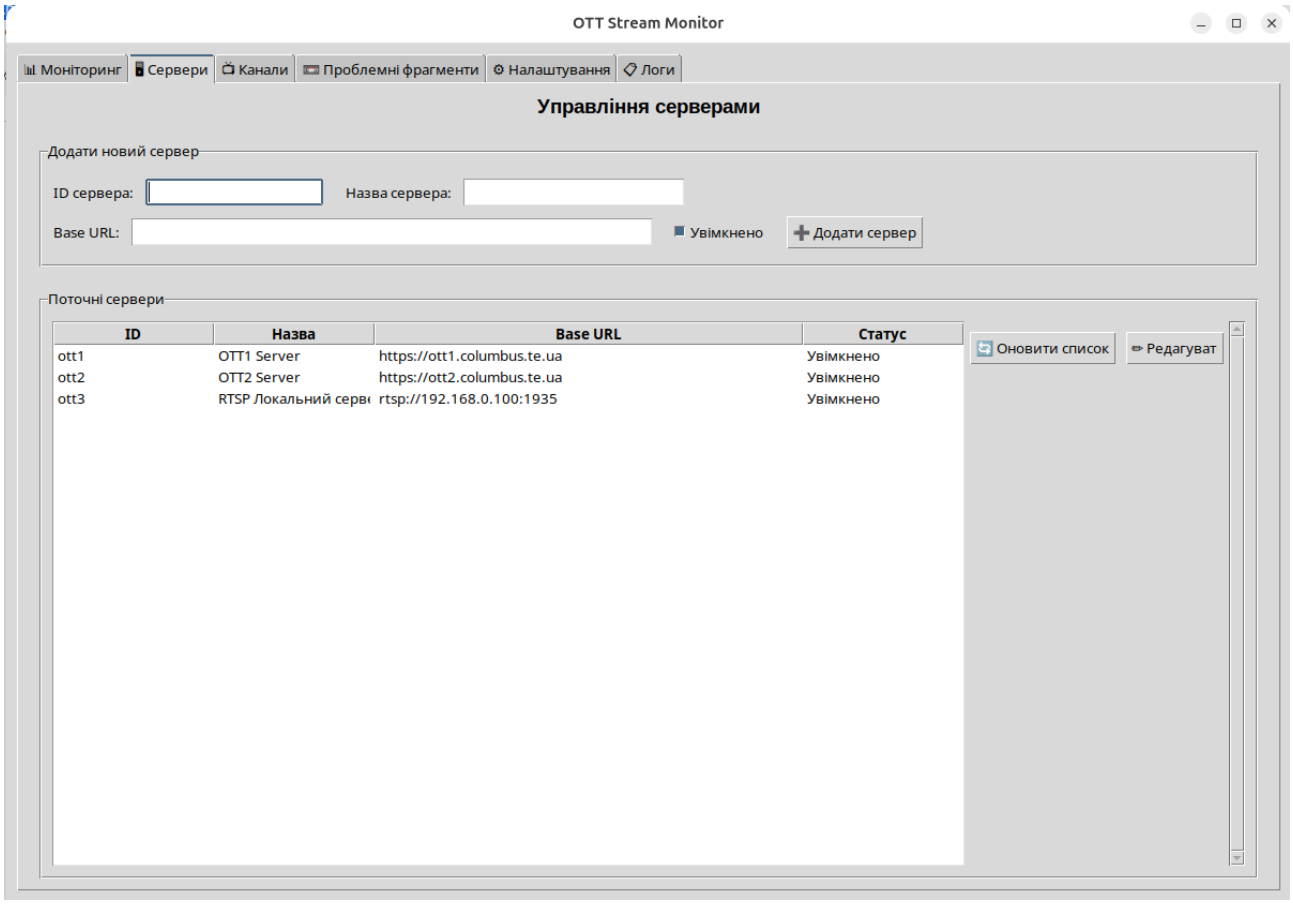

Рисунок 3.1 - Вікно управління серверами.

Після налаштування серверів переходимо до управління каналами на вкладці "Канали". Тут можна додавати окремі канали вручну або імпортувати їх списки з файлів чи буфера обміну. Для додавання окремого каналу обираємо сервер зі списку, вводимо назву каналу та його URL. Система підтримує різні формати URL, включаючи HLS плейлисти (.m3u8), RTSP потоки та HTTP посилання.

Масове додавання каналів можливе через функцію "Додати з тексту", де кожен рядок має містити назву каналу та URL, розділені символом вертикальної риски, табуляції або коми. Також можна скопіювати список каналів в буфер обміну та використати функцію "Додати з буфера обміну" для швидкого імпорту.

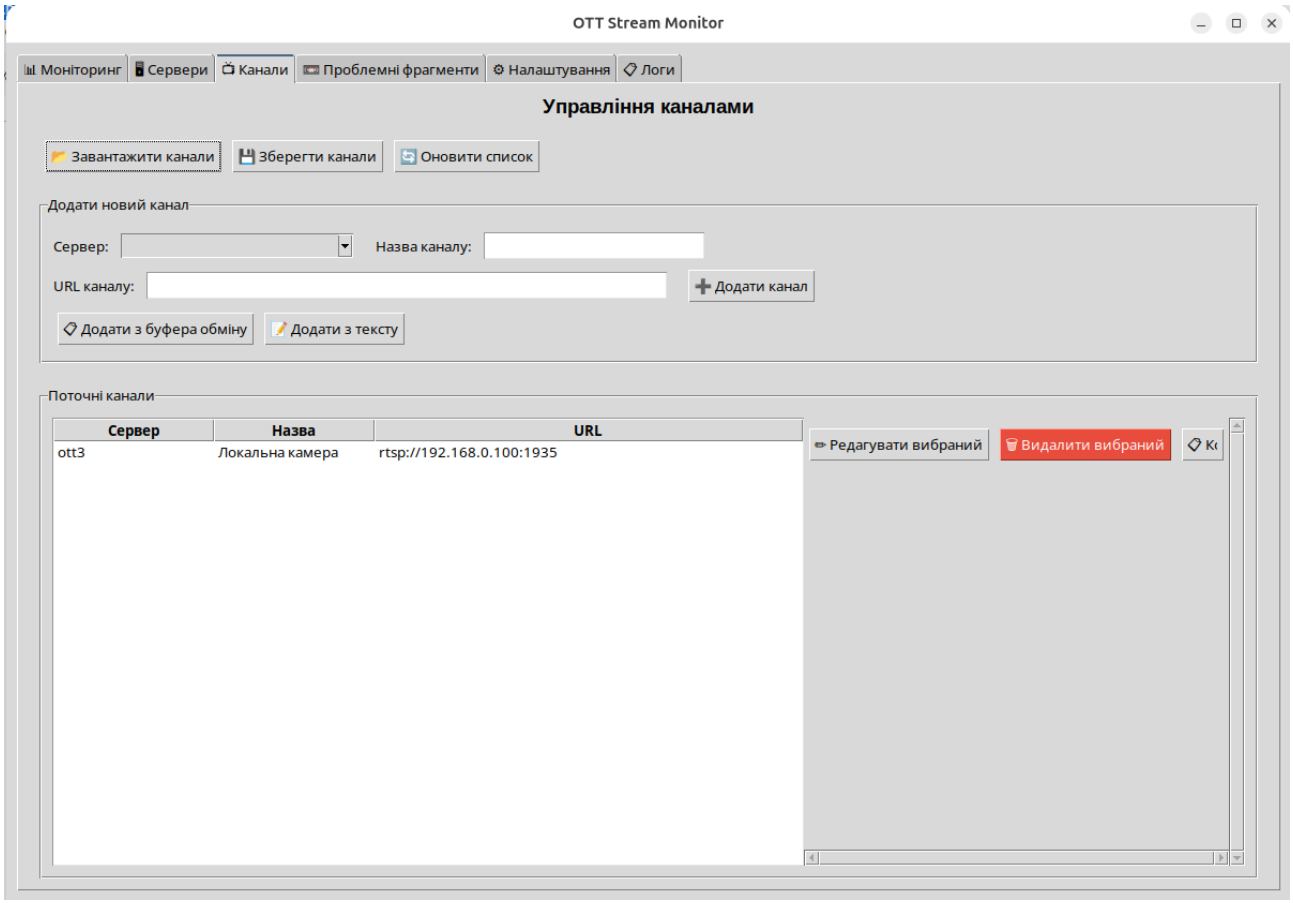

Рисунок 3.2 - Вікно управління каналами

Налаштування параметрів моніторингу здійснюється на вкладці "Налаштування". Тут можна змінити інтервал перевірки каналів, максимальну кількість одночасних потоків для аналізу, тривалість зразків для аналізу, пороги для виявлення проблем звуку та синхронізації, а також URL вебхука для Google Chat сповіщень. Рекомендується встановити інтервал перевірки не менше 300 секунд для забезпечення стабільної роботи системи.

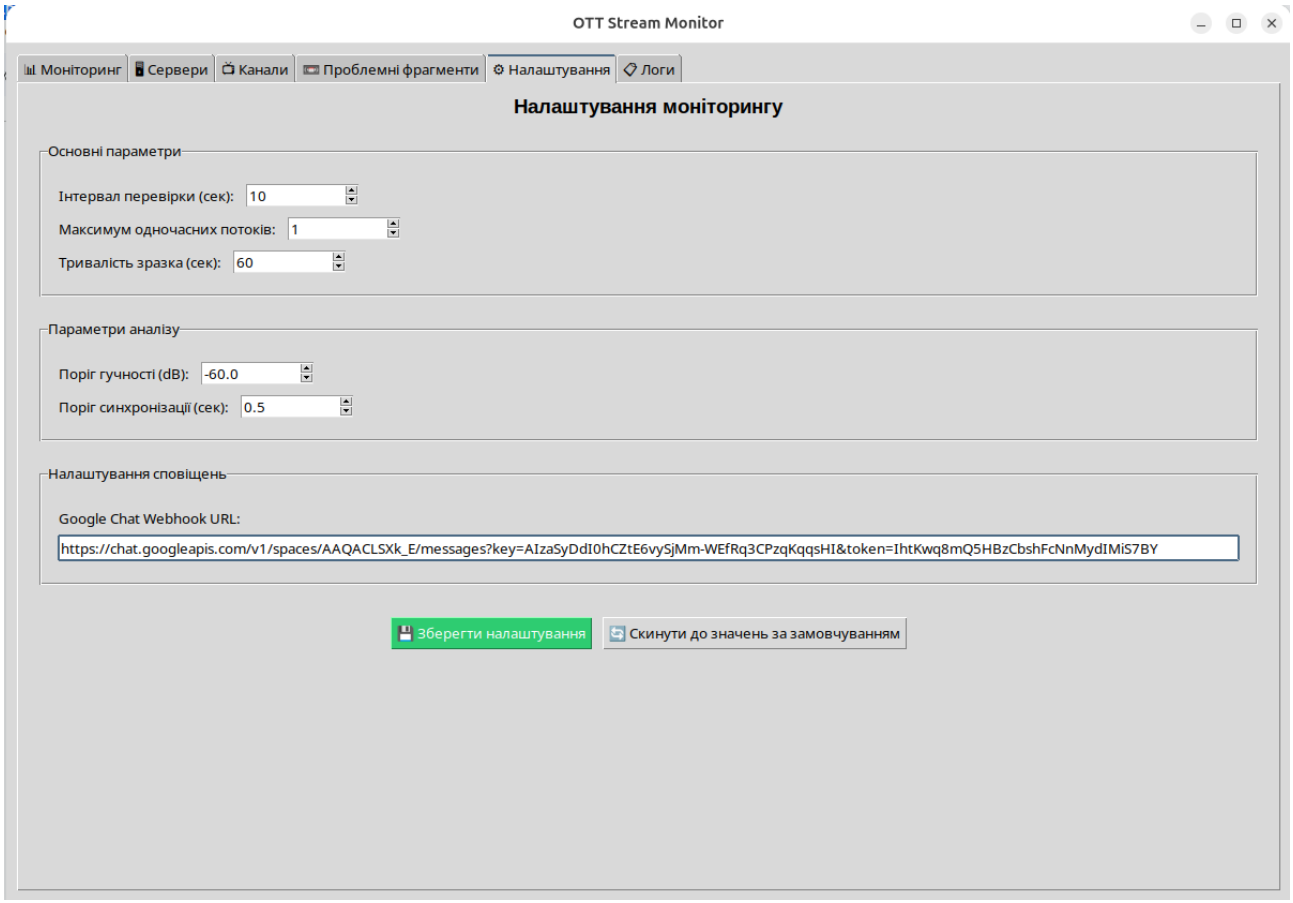

Рисунок 3.3 - Вікно налаштувань.

Основна робота з системою відбувається на вкладці "Моніторинг". Тут відображається поточний статус системи, список каналів для моніторингу та панель управління. Для запуску моніторингу спочатку необхідно обрати сервери, які потрібно контролювати, за допомогою відповідних прапорців. Після вибору серверів натискаємо кнопку "Запустити моніторинг", і система почне циклічну перевірку всіх каналів на обраних серверах.

Попередній перегляд відео дозволяє візуально оцінити якість потоку в реальному часі. Для його активації необхідно вибрати канал у списку та натиснути кнопку "Старт" в панелі попереднього перегляду. Система автоматично визначить тип потоку та налаштує оптимальні параметри підключення.

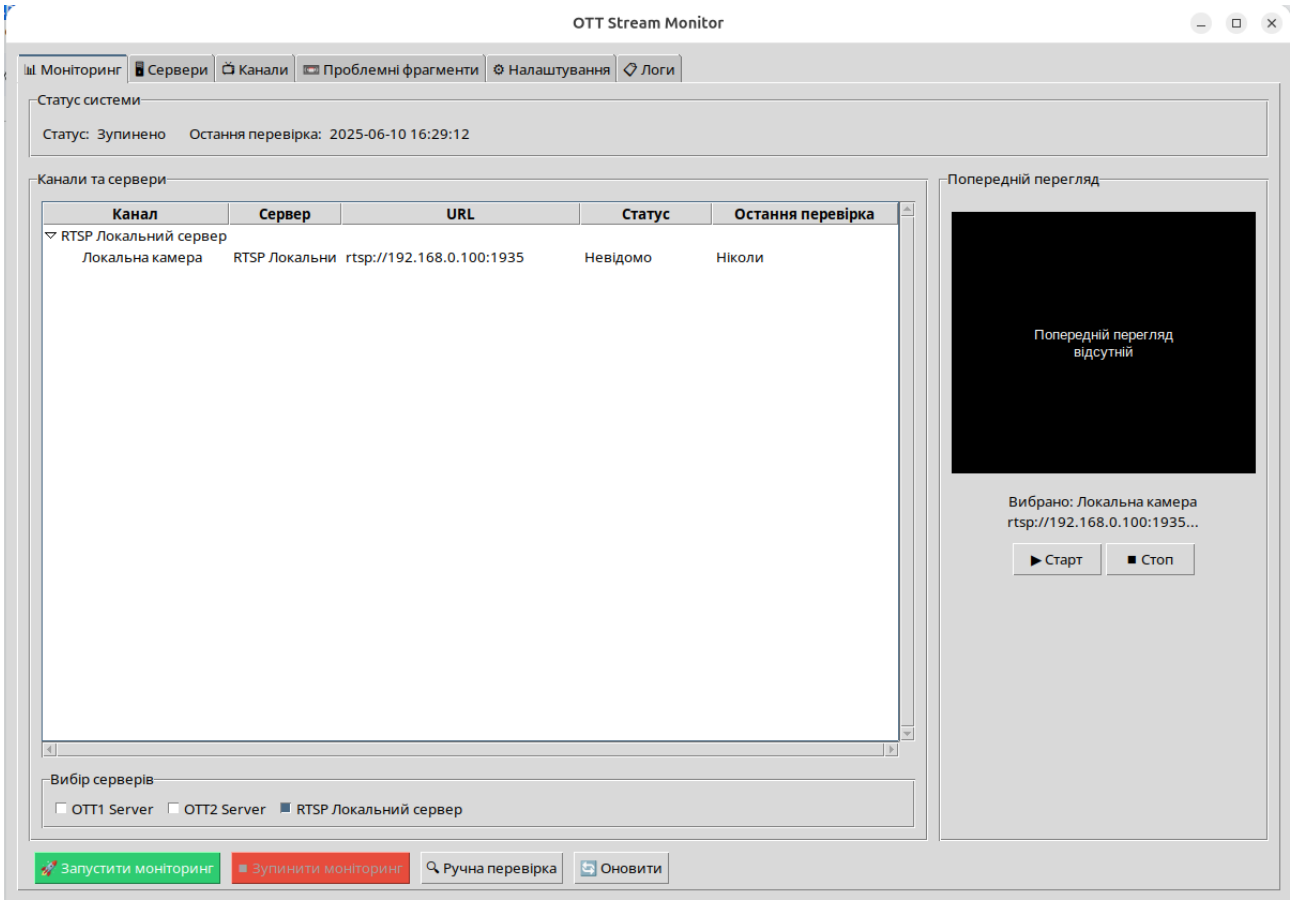

Рисунок 3.4 - Вікно моніторингу

Система автоматично виявляє різні типи проблем потоків, включаючи відсутність звуку, проблеми синхронізації аудіо та відео, чорні екрани, зависання відео та недоступність потоків. При виявленні проблем система зберігає фрагменти з помилками у папці "error_fragments" для подальшого аналізу. Ці фрагменти можна переглянути на вкладці "Проблемні фрагменти".

Вкладка "Проблемні фрагменти" містить список всіх збережених фрагментів з проблемами, відсортований за датою створення. Кожен фрагмент можна відтворити за допомогою системного відеоплеєра для детального аналізу проблеми або видалити після вирішення питання.

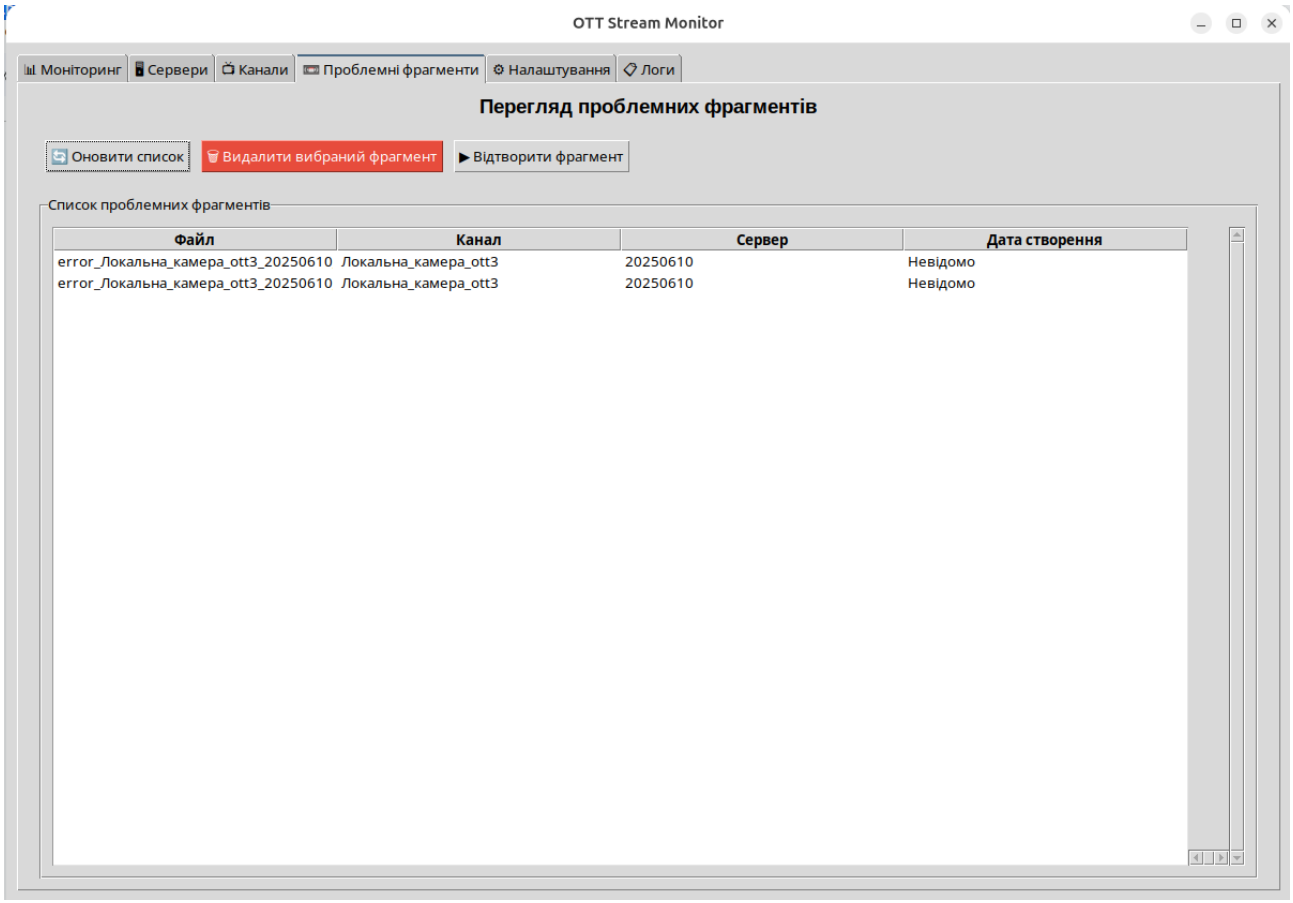

Рисунок 3.5 - Вікно перегляду проблемних фрагментів

Система сповіщень автоматично надсилає повідомлення про виявлені проблеми в Google Chat через налаштований вебхук. Повідомлення групуються по каналах та серверах, а також організовуються в треди для зручного відстеження процесу вирішення проблем. При вирішенні проблеми система автоматично надсилає повідомлення про відновлення нормальної роботи каналу.

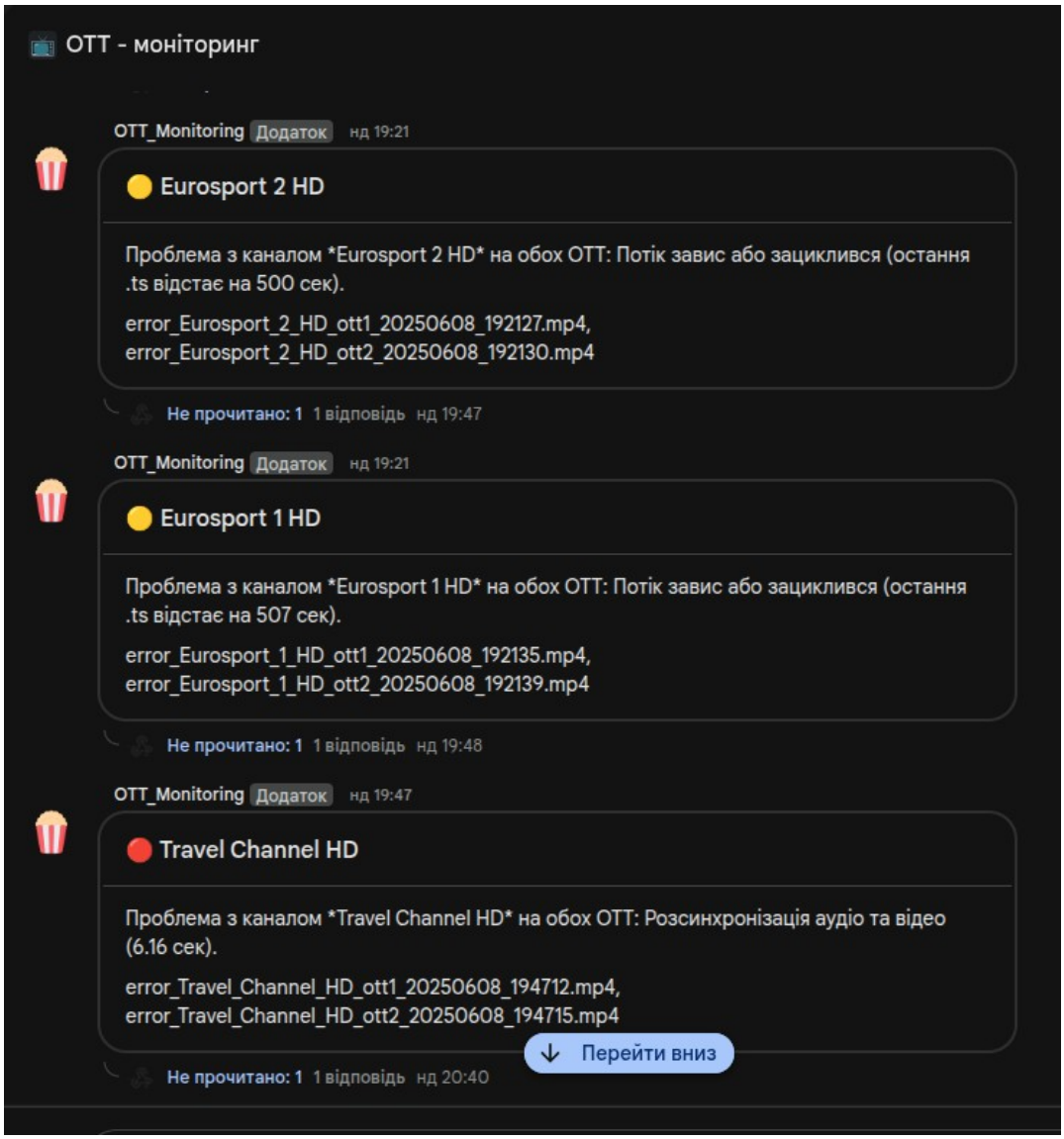

Рисунок 3.6 - Вікно перегляду проблемних фрагментів.

Перегляд логів системи доступний на відповідній вкладці, де відображається детальна інформація про всі операції системи, включаючи результати перевірок, виявлені проблеми та системні події. Логи можна очистити або зберегти у файл для подальшого аналізу.

Для зупинки моніторингу використовується кнопка "Зупинити моніторинг" на головній вкладці. Система коректно завершить всі активні перевірки та збереже поточний стан для можливості подальшого відновлення роботи.

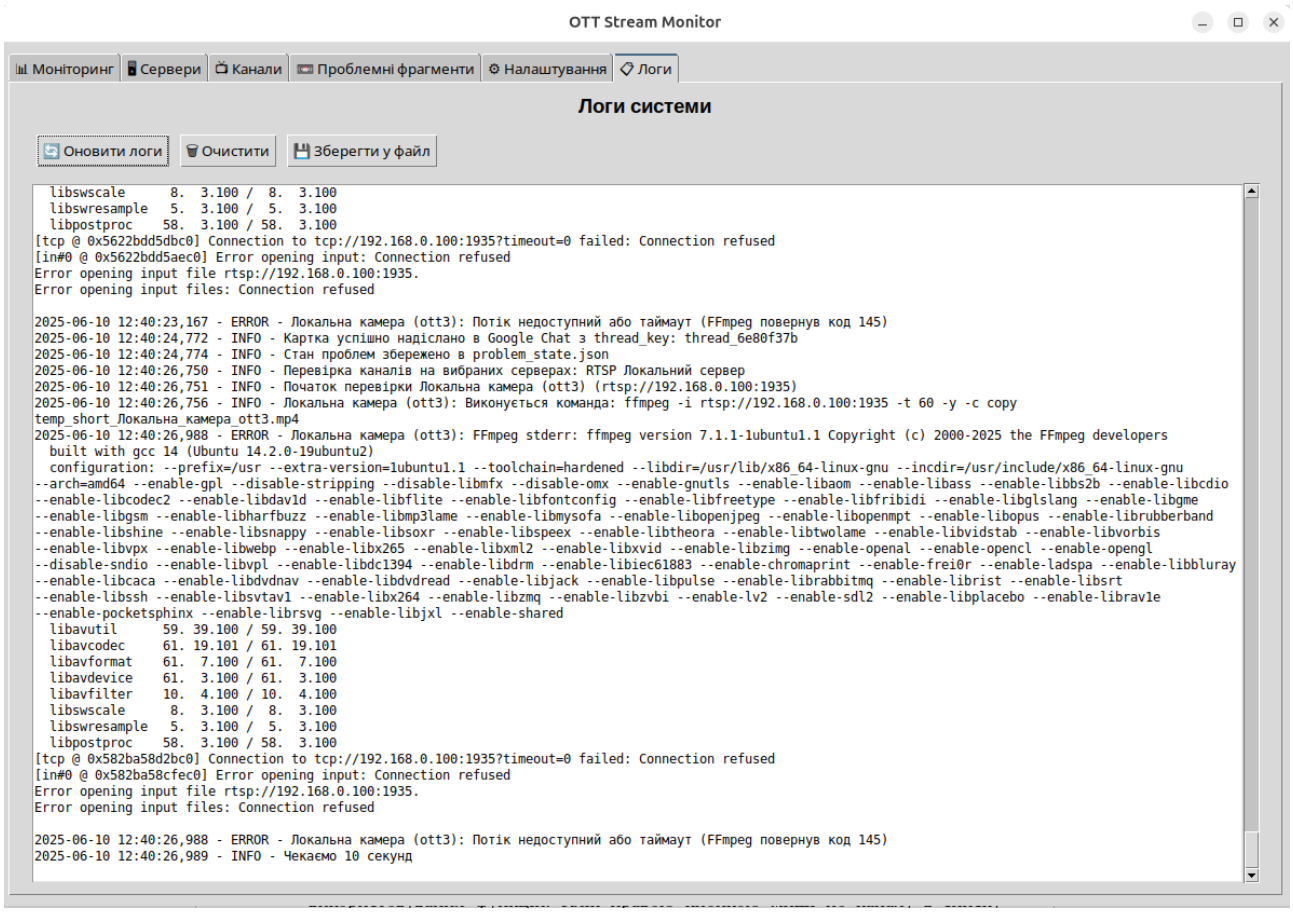

Рисунок 3.7 - Вікно перегляду проблемних фрагментів.

Експорт та імпорт конфігурації каналів здійснюється через відповідні кнопки на вкладці "Канали". Це дозволяє легко переносити налаштування між різними інсталяціями системи або створювати резервні копії конфігурації. Файли конфігурації зберігаються у форматі JSON та можуть редагуватися зовнішніми текстовими редакторами при необхідності.

Система також підтримує контекстні меню для швидкого доступу до часто використовуваних функцій. Клік правою кнопкою миші по каналу в списку відкриває меню з опціями редагування, видалення, копіювання URL та відкриття посилання в браузері.

ВИСНОВКИ

Розробку системи мережевого моніторингу OTT-потоків можна умовно поділити на чотири основні етапи, кожен з яких вніс суттєвий внесок у створення комплексного та функціонального рішення для автоматизованого контролю якості медіа-потоків з урахуванням мережевих технологій та протоколів.

Перший етап полягав у ретельному аналізі предметної області з акцентом на мережевих аспектах та формуванні network-oriented вимог до системи. На цьому етапі було проведено глибоке дослідження специфіки OTT мережевих технологій та виявлено ключові мережеві проблеми, які виникають при трансляції медіа-контенту через різні network protocols та CDN infrastructures. Особлива увага приділялася аналізу типових мережевих сценаріїв використання системи операторами, включаючи network quality monitoring, виявлення network-specific technical problems, управління мережевою конфігурацією каналів та обробку network emergency situations.

Важливою частиною цього етапу став аналіз існуючих мережевих рішень на ринку систем моніторингу медіа-потоків. Порівняння різних network-oriented підходів до вирішення завдань контролю якості дозволило виявити переваги та недоліки наявних мережевих систем, що стало основою для формування унікальних network solutions, які увійшли до розробленого проєкту. Зокрема, було виявлено недостатню увагу існуючих систем до інтеграції штучного інтелекту для аналізу мережевих проблем та автоматизації прийняття рішень в network context.

Другий етап був присвячений проектуванню мережевої архітектури програмної системи та створенню детальних технічних специфікацій з урахуванням network performance requirements. Цей етап розпочався з розробки distributed network-aware architecture, яка забезпечує високу мережеву гнучкість та можливість horizontal scaling. Було обрано архітектурний підхід, що поєднує local network processing з можливістю distributed monitoring множини network

endpoints, що дозволяє ефективно контролювати великі медіа-інфраструктури через optimized network connections.

Ключовим рішенням на етапі проектування стало створення adaptive network configuration management system, яка автоматично адаптується до різних типів медіа-потоків та network transmission protocols. Це рішення значно спрощує процес налаштування мережевої системи та забезпечує її сумісність з широким спектром OTT платформ, CDN providers та network technologies.

Третій етап охоплював програмну реалізацію всіх мережевих компонентів системи з використанням сучасних network technologies та найкращих практик розробки network-aware програмного забезпечення. Реалізація розпочалася з створення ядра системи network stream analysis, яке інтегрує потужні можливості FFmpeg для детального аналізу якості медіа-контенту з урахуванням network conditions. Було розроблено складні алгоритми виявлення різних типів network-induced problems, включаючи аналіз network latency impact, packet loss detection, bandwidth degradation analysis та protocol-specific issues.

Революційною особливістю реалізації стала інтеграція великої мовної моделі для інтелектуального аналізу мережевих проблем та автоматичного створення network troubleshooting recommendations. Ця функціональність значно підвищує ефективність роботи network operators, надаючи контекстуальний аналіз мережевих проблем та персоналізовані інструкції для їх вирішення з урахуванням network topology та protocol specifics.

Четвертий етап включав комплексне мережеве тестування системи та підготовку до промислової експлуатації в production network environments. Було проведено багаторівневе network testing, включаючи network unit testing окремих компонентів, integration testing взаємодії між мережевими модулями, functional testing всіх network scenarios та comprehensive load testing для визначення network performance characteristics під різними network conditions.

Результати мережевого тестування підтвердили високу надійність та ефективність розробленої network-oriented системи. Network unit testing виявило

відсутність критичних помилок в окремих мережевих компонентах, а integration testing показало злагоджену роботу всіх network subsystems з мінімальною latency та максимальною throughput optimization.

У результаті виконання всіх етапів розробки було створено інноваційну систему мережевого моніторингу OTT-потоків, яка поєднує традиційні методи аналізу медіа-контенту з сучасними мережевими технологіями та штучним інтелектом. Система забезпечує автоматизований контроль мережевої якості потоків, інтелектуальний аналіз network-specific problems та ефективну підтримку операторів у вирішенні мережевих технічних завдань, що значно підвищує надійність та якість медіа-сервісів в distributed network environments.

Створена система демонструє високу ефективність в роботі з різними мережевими протоколами (HTTP/HTTPS, RTSP/TCP/UDP, WebRTC), забезпечує intelligent network load balancing, automatic failover mechanisms та comprehensive network diagnostics, що робить її незамінним інструментом для modern OTT service providers, які потребують reliable network-aware media quality monitoring.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пукас А.В., Шевчук Р.П., Крепич С.Я. Методичні вказівки до виконання та захисту випускної кваліфікаційної роботи здобувачів вищої освіти рівня «бакалавр» за спеціальністю 126 «Інформаційні системи та технології» освітньо-професійної програми «Інформаційні системи та технології», 2020. URL: http://dspace.wunu.edu.ua/bitstream/316497/42183/1/Bak_DP_2020.pdf (дата звернення: 06.06.2024)
2. ITU-T Recommendation H.264: Advanced video coding for generic audiovisual services / International Telecommunication Union. – Geneva: ITU-T, 2019. – 812 p.
3. RFC 3550: RTP: A Transport Protocol for Real-Time Applications / H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson // Internet Engineering Task Force. – 2003. – 89 p.
4. RFC 2326: Real Time Streaming Protocol (RTSP) / H. Schulzrinne, A. Rao, R. Lanphier // Internet Engineering Task Force. – 1998. – 92 p.
5. HTTP Live Streaming (HLS) Specification / Apple Inc. – Cupertino: Apple Developer Documentation, 2021. – 156 p.
6. Dynamic Adaptive Streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats / ISO/IEC 23009-1:2019. – Geneva: International Organization for Standardization, 2019. – 234 p.
7. FFmpeg Documentation: A complete, cross-platform solution to record, convert and stream audio and video [Електронний ресурс] / FFmpeg Developers. – Режим доступу: https://ffmpeg.org/documentation.html. – Дата доступу: 15.11.2024.
8. Anthropic Claude API Documentation [Електронний ресурс] / Anthropic. – Режим доступу: https://docs.anthropic.com/. – Дата доступу: 20.11.2024.
9. Python Software Foundation. Python 3.9 Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3.9/. – Дата доступу: 10.11.2024.
10. OpenCV: Open Source Computer Vision Library Documentation [Електронний ресурс] / OpenCV Team. – Режим доступу: https://docs.opencv.org/. – Дата

доступу: 12.11.2024.

11. Tkinter Documentation: Python GUI Programming [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://docs.python.org/3/library/tkinter.html. – Дата доступу: 08.11.2024.

12. Quality of Experience (QoE) for multimedia services: Subjective methods for assessing quality / ITU-T Recommendation P.910. – Geneva: ITU-T, 2021. – 67 p.

13. Stockhammer, T. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles / T. Stockhammer // Proceedings of the 2nd Annual ACM Conference on Multimedia Systems. – New York: ACM, 2011. – P. 133-144.

14. Zambelli, A. IIS Smooth Streaming Technical Overview / A. Zambelli // Microsoft Technical Documentation. – Redmond: Microsoft Corporation, 2009. – 45 p.

15. Seufert, M. A Survey on Quality of Experience of HTTP Adaptive Streaming / M. Seufert, S. Egger, M. Slanina // IEEE Communications Surveys & Tutorials. – 2015. – Vol. 17, No. 1. – P. 469-492.

16. Вишневський, В.М. Мережі телекомунікацій: основи теорії та застосування / В.М. Вишневський, О.І. Ляценко, С.О. Гаркуша. – Київ: Техніка, 2019. – 512 с.

17. Van Rossum, G. Python Tutorial / G. Van Rossum, F.L. Drake Jr. – Amsterdam: Python Software Foundation, 2021. – 156 p.

18. Requests: HTTP for Humans Documentation [Електронний ресурс] / Kenneth Reitz. – Режим доступу: https://docs.python-requests.org/. – Дата доступу: 14.11.2024.

19. Docker Documentation: Containerization Platform [Електронний ресурс] / Docker Inc. – Режим доступу: https://docs.docker.com/. – Дата доступу: 18.11.2024.

20. Google Chat API Reference [Електронний ресурс] / Google LLC. – Режим доступу: https://developers.google.com/chat/api/. – Дата доступу: 16.11.2024.

21. JSON Schema Specification [Електронний ресурс] / JSON Schema Team. – Режим доступу: https://json-schema.org/specification.html. – Дата доступу: 13.11.2024.

22. Fielding, R.T. Architectural Styles and the Design of Network-based Software Architectures: Doctoral dissertation / R.T. Fielding. – Irvine: University of California, 2000. – 162 p.

23. Sommerville, I. Software Engineering / I. Sommerville. – 10th edition. – Boston: Pearson, 2016. – 816 p.

24. Beck, K. Test-Driven Development: By Example / K. Beck. – Boston: Addison-Wesley Professional, 2003. – 240 p.

25. Martin, R.C. Clean Code: A Handbook of Agile Software Craftsmanship / R.C. Martin. – Upper Saddle River: Prentice Hall, 2008. – 464 p.

26. Gamma, E. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. – Reading: Addison-Wesley, 1995. – 395 p.

27. PIL (Python Imaging Library) Documentation [Електронний ресурс] / Alex Clark and Contributors. – Режим доступу: https://pillow.readthedocs.io/. – Дата доступу: 11.11.2024.

28. Threading in Python Documentation [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://docs.python.org/3/library/threading.html. – Дата доступу: 09.11.2024.

29. Logging facility for Python [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://docs.python.org/3/library/logging.html. – Дата доступу: 07.11.2024.