МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГОТЯШ Юрій Павлович

Метод вибору стрімінгових протоколів RTMP та DASH за допомогою машинного навчання. / Method for Selecting Streaming Protocols RTMP and DASH Using Machine Learning

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
Ю.П. Готяш
_____

Науковий керівник:
к.т.н., доцент П.Є. Биковий
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.М. Васильків

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Готяш Юрій Павлович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Метод вибору стрімінгових протоколів RTMP та DASH за допомогою машинного навчання / Method for Selecting Streaming Protocols RTMP and DASH Using Machine Learning
керівник роботи к.т.н., доцент П.Є. Биковий,
затверджені наказом по університету від 12 грудня 2023 року № 753.

2. Строк подання студентом закінченої кваліфікаційної роботи 4 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- огляд предметної області стрімінгових стрімінгових сервісів і протоколів;
- аналіз методів багатокритеріального оцінювання;
- огляд існуючих рішень, що впливають на продуктивність стрімінгових систем;
- постановка завдання дослідження;
- розробка методології багатокритеріального аналізу стрімінгових протоколів;
- опис обробки та методів обробки використаних даних;
- реалізація методу вибору стрімінгових протоколів;
- тестування та оцінка ефективності запропонованого підходу;

5. Перелік графічного матеріалу у роботі
- схема архітектури системи оцінки продуктивності стрімінгових протоколів;
- графіки результатів порівняння продуктивності стрімінгових протоколів.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 30.11.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2024 р.	

Студент _____ Ю.П. Готяш
підпис

Керівник роботи _____ к.т.н., доцент П.Є. Биковий
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод вибору стрімінгових протоколів RTMP та DASH за допомогою машинного навчання» виконана для здобуття освітнього ступеня «Магістр» за спеціальністю 122 «Комп'ютерні науки» в межах освітньо-професійної програми «Комп'ютерні науки» написана обсягом 67 сторінок і містить 8 ілюстрацій, 4 додатки та 38 використаних джерела.

Метою роботи є створення методу вибору стрімінгових протоколів за допомогою машинного навчання, що дозволяє підвищити продуктивність систем потокового відео шляхом об'єктивного вибору протоколу передачі даних, який оптимально відповідає заданим умовам роботи та вимогам до якості обслуговування.

В роботі використані наступні методи машинного навчання: авторегресійна інтегрована модель ковзного середнього, байєсівська регресія хребта, регресія з градієнтним бустингом, регресія опорних векторів, довготривала короткочасна пам'ять.

Результати дослідження: Кваліфікаційна робота включає розробку методу вибору стрімінгового протоколу на основі багатокритеріального аналізу на базі Ant Media Server, що дозволяє об'єктивно оцінювати продуктивність протоколів передачі даних за сукупністю технічних критеріїв, для забезпечення оптимальної роботи систем потокового відео в різних умовах експлуатації.

Практичне значення роботи полягає в розробці методу для оптимального вибору стрімінгового протоколу який дозволяє забезпечити ефективне управління ресурсами систем потокового відео та підвищити якість обслуговування користувачів.

Рекомендації для подальших досліджень полягають у вдосконаленні моделей прогнозування навантаження шляхом інтеграції нових методів машинного навчання.

Ключові слова: ANT MEDIA SERVER, АНАЛІЗ ДАНИХ, ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ, АВТОМАТИЗАЦІЯ АНАЛІЗУ.

ABSTRACT

The qualification work on the topic «Method for Selecting Streaming Protocols RTMP and DASH Using Machine Learning» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 67 pages and it contains 8 figures, 4 annexes and 38 sources.

The aim of the study is to create a multi-criteria analysis method for evaluating streaming protocols using machine learning methods, which allows to improve the performance of video streaming systems by objectively selecting a data transmission protocol that optimally meets the specified operating conditions and quality of service requirements.

The following machine learning methods were used in the work: Auto Regression Integrated Moving Average (ARIMA), Bayesian Ridge Regression (BRR), Gradient Boosting Regressor (GBR), Support Vector Regression (SVR), and Long Short-Term Memory (LSTM).

Research results: The qualification work includes the development of a method for selecting a streaming protocol based on multicriteria analysis based on Ant Media Server, which allows to objectively evaluate the performance of data transmission protocols according to a set of technical criteria to ensure optimal operation of video streaming systems in various operating conditions.

The practical significance of the work is to develop a method for the optimal selection of a streaming protocol that allows for efficient resource management of video streaming systems and improve the quality of user service.

Recommendations for further research are to improve load prediction models by integrating new machine learning methods.

Keywords: ANT MEDIA SERVER, DATA ANALYSIS, FORECASTING, MACHINE LEARNING, ANALYSIS AUTOMATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES-128 - Advanced Encryption Standard (128-bit)

AI - Artificial Intelligence

CDN - Content Delivery Network

CSV - Comma-Separated Values

DASH - Dynamic Adaptive Streaming over HTTP

DRM - Digital Rights Management

HLS - HTTP Live Streaming

HTTP - Hypertext Transfer Protocol

LSTM - Long Short-Term Memory

MPD - Media Presentation Description

MPEG-DASH - Moving Picture Experts Group Dynamic Adaptive Streaming over HTTP

NAT - Network Address Translation

QoE - Quality of Experience

RTMP - Real-Time Messaging Protocol

SRT - Secure Reliable Transport

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

VOD - Video on Demand

WebRTC - Web Real-Time Communication

XML - Extensible Markup Language

ВСТУП

Стрімінгові сервіси займають ключове місце в індустрії цифрових технологій, надаючи користувачам доступ до контенту в реальному часі. Зростання популярності відео- та аудіо-стрімінгу вимагає від провайдерів високої ефективності та стабільності платформ. Для забезпечення якісного стрімінгу в умовах високого навантаження необхідна складна інфраструктура, здатна динамічно адаптуватися до змінюваного попиту. Важливим аспектом є наявність потужних апаратних ресурсів, що можуть обробляти великі обсяги даних, включаючи відео- та аудіопотоки, а також забезпечувати безперебійне транслювання контенту для великої кількості користувачів. Для оцінки продуктивності систем потокового відео важливо вибрати оптимальний стрімінговий протокол, враховуючи специфічні умови роботи. Традиційні методи аналізу, такі як технічний і фундаментальний аналіз, часто ґрунтуються на історичних даних і вимагають значних витрат часу та ресурсів для досягнення точних результатів. Водночас сучасні методи машинного навчання, зокрема багатокритеріальний аналіз, дозволяють ефективно вирішувати ці завдання з меншими витратами часу і ресурсів. Це відкриває нові можливості для автоматизації управління ресурсами в стрімінгових сервісах, підвищуючи точність прогнозів і забезпечуючи оптимальну продуктивність при динамічних змінах умов навантаження.

Метою даного дослідження є розробка та тестування методу вибору стрімінгового протоколу для платформ потокового відео на основі прогнозування умов мережі за допомогою машинного навчання. Завдяки цьому методу планується оптимізувати вибір протоколу в реальному часі, враховуючи змінні умови мережі, такі як затримка, пропускна здатність та втрати пакетів. Це дозволить забезпечити стабільну роботу стрімінгових платформ при змінному навантаженні, підвищити ефективність передачі контенту та покращити якість обслуговування для користувачів.

Метою цього дослідження є розробка методу на основі машинного навчання, який зможе автоматично вибирати оптимальний протокол для відео- та аудіотрансляцій, враховуючи мережеві умови, типи пристроїв користувачів та вимоги до якості трансляцій, забезпечуючи оптимальний баланс між швидкістю передачі та якістю контенту. Цей алгоритм інтегрується з Ant Media Server, що дозволить протестувати рішення в реальних умовах

Об’єктом дослідження є інфраструктура стрімінгових сервісів, зокрема процеси вибору та оцінки стрімінгових протоколів для забезпечення високої продуктивності системи потокового відео. Дослідження охоплює аналіз та порівняння різних протоколів з урахуванням багатьох критеріїв, таких як затримка, пропускна здатність, якість обслуговування (QoE) та енергоспоживання, для визначення оптимального варіанту в конкретних умовах мережі та навантаження.

Предметом дослідження є алгоритми та методи, що застосовуються для прогнозування навантаження на стрімінгові системи, а також алгоритми автоматичного вибору стрімінгових протоколів для забезпечення оптимальної продуктивності систем потокового відео на базі Ant Media Server. Дослідження охоплює оцінку ефективності різних протоколів за допомогою багатокритеріального аналізу для забезпечення стабільної роботи стрімінгових платформ при змінних умовах мережі та навантаження.

Для досягнення поставленої мети були використані такі методи дослідження, як аналіз існуючих рішень для автоматизації вибору стрімінгових протоколів, методи машинного навчання для прогнозування навантаження на стрімінгові системи, а також методи тестування програмного забезпечення для оцінки ефективності реалізованих алгоритмів вибору протоколів і оптимізації продуктивності платформ.

Наукова новизна дослідження полягає в розробці методу вибору стрімінгового протоколу. Метод використовує машинне навчання для прогнозування навантаження на стрімінгові системи та здійснює адаптивний вибір протоколу в залежності від поточних параметрів мережі, що дозволяє

забезпечити стабільну та ефективну роботу платформ потокового відео з мінімальними витратами ресурсів і високою якістю обслуговування користувачів.

Наукова новизна дослідження полягає в розробці методу вибору стрімінгового протоколу на основі машинного навчання. Запропонований метод прогнозує навантаження на стрімінгові системи та здійснює вибір протоколу відповідно до поточних параметрів мережі. Це забезпечує стабільну й ефективну роботу платформ потокового відео, мінімізуючи витрати ресурсів і підтримуючи високу якість обслуговування користувачів

Практичне значення роботи полягає в розробці методу автоматизованого вибору стрімінгового протоколу для сервісів на базі Ant Media Server, з використанням методів машинного навчання для прогнозування навантаження на систему.

Дане дослідження дозволяє використати інтелектуальні рішення у сфері потокового передавання та використання адаптивних протоколів, що можуть застосовуватися не лише у стрімінгу, але й у веб-конференціях, вебінарах та інших інноваційних сервісах.

Апробація результатів дослідження здійснювалася шляхом представлення основних теоретичних положень і практичних результатів на міжнародній мультидисциплінарній науковій інтернет-конференції “Світ наукових досліджень” (м. Тернопіль, Україна), а також на Міжнародній науково-практичній інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (м.Тернопіль, Україна).

1 ТЕОРЕТИКО-МЕТОДИЧНІ ЗАСАДИ АНАЛІЗУ СТРІМІНГОВИХ ПРОТОКОЛІВ

1.1 Аналіз предметної області

Потокове передавання медіа — це сучасний спосіб доставки аудіо- та відеоконтенту користувачам у режимі реального часу, без необхідності попереднього завантаження файлів. Термін «потік» відображає процес, у якому користувач отримує доступ до контенту одразу після початку передачі даних від сервера. На відміну від традиційного завантаження файлів, під час потокового передавання медіа можна миттєво споживати — наприклад, дивитися фільм чи слухати музику в реальному часі.

Потокове передавання стало ключовим елементом сучасних інтернет-сервісів, оскільки дозволяє користувачам отримувати доступ до великого обсягу мультимедійного вмісту без зберігання його на локальних пристроях. YouTube, Netflix і Twitch є одними з найбільших і найпопулярніших платформ для потокового відео, і кожна з них має свої особливості в управлінні потоками відео та виборі стрімінгових протоколів [1-3].

YouTube використовує адаптивне стрімінгове рішення за допомогою протоколу DASH (Dynamic Adaptive Streaming over HTTP), що дозволяє регулювати якість відео в реальному часі, залежно від доступної пропускної здатності мережі користувача. Цей підхід забезпечує безперервний перегляд відео з високою роздільною здатністю, мінімізуючи буферизацію навіть при нестабільних мережевих умовах. Для живих трансляцій YouTube також використовує RTMP (Real-Time Messaging Protocol), що дозволяє забезпечити мінімальну затримку в прямому ефірі [4].

Netflix, як один з лідерів у сфері відео-стрімінгу, також використовує DASH для адаптивного стрімінгу, що дозволяє оптимізувати якість відео в залежності від доступної пропускної здатності мережі користувача. Netflix активно застосовує технології для мінімізації буферизації, що є критичним для користувацького досвіду, особливо для перегляду контенту в 4K та Ultra HD. Ця

адаптивність дозволяє Netflix ефективно передавати відео навіть за умов різних мережевих навантажень.

Twitch, популярна платформа для ігрових трансляцій, використовує RTMP для забезпечення низької затримки в прямих ефірах, що є важливим для інтерактивних ігрових стрімів. Платформа також має механізми для автоматичного підстроювання якості потоку в залежності від мережевих умов за допомогою HLS (HTTP Live Streaming). Це дозволяє забезпечити безперервний перегляд відео при змінних пропускних здатностях, мінімізуючи проблеми з буферизацією.

Усі три платформи активно використовують різні технології для адаптації якості відео в реальному часі, забезпечуючи високий рівень задоволення користувачів, знижуючи буферизацію і забезпечуючи стабільність потоку навіть у разі змінних умов мережі. Крім цих платформ, існують різні технології для потокового передавання, кожна з яких має свої особливості та цілі. Серед основних технологій виділяють наступні: RTMP (Real-Time Messaging Protocol) — використовується для живих трансляцій з низькою затримкою; DASH (Dynamic Adaptive Streaming over HTTP) — забезпечує адаптацію якості потоку до пропускної здатності мережі; HLS (HTTP Live Streaming) — популярний протокол для трансляцій на мобільних пристроях; WebRTC (Web Real-Time Communication) — дозволяє миттєві відеоз'єднання та живі трансляції з мінімальною затримкою, часто застосовується в веб-конференціях та ігрових стрімах; SRT (Secure Reliable Transport) — забезпечує надійну передачу медіаданих навіть у нестабільних мережах [5].

Ant Media Server — це платформа для трансляції відео в реальному часі, яка підтримує кілька популярних протоколів [6], таких як RTMP, HLS (HTTP Live Streaming), DASH (Dynamic Adaptive Streaming over HTTP), а також WebRTC для низькозатратної передачі в реальному часі. Цей сервер оптимізовано для забезпечення високоякісного відео стрімінгу з низькою затримкою, що робить його ідеальним для таких додатків, як відеоконференції, ігрові стріми, навчальні платформи та інші інтерактивні трансляції.

Крім того, Ant Media Server активно підтримує масштабованість через кластеризацію, що дає змогу обробляти величезні обсяги відеопотоків одночасно, забезпечуючи стабільну роботу навіть при високих навантаженнях. Це особливо важливо для великих подій, таких як прямі трансляції або міжнародні конференції, де необхідно забезпечити ефективну доставку контенту на різні пристрої по всьому світу.

Завдяки зростанню попиту на потокові сервіси з мінімальною затримкою, важливою є здатність систем адаптуватися до змін у пропускній здатності мережі. Використання Ant Media Server знижує складність інтеграції кількох протоколів і дає змогу автоматизувати вибір за допомогою машинного навчання, що робить це рішення перспективним для застосування в реальних продуктах.

Перший етап роботи полягав в ознайомленні зі стрімінговими платформами, які вже використовуються. Важливо було вивчити, як працюють ці платформи та які протоколи вони використовують для передавання медіа в реальному часі. Окрему увагу приділяли аналізу можливостей таких технологій, як RTMP та DASH, а також перспективам впровадження WebRTC для забезпечення інтерактивних живих трансляцій із мінімальною затримкою.

Наступним важливим кроком стало вивчення можливостей інтеграції Ant Media Server у внутрішні проекти компанії. Ant Media Server виявився перспективним рішенням для підтримки низьколатентних стрімінгів і адаптивної передачі відео. Це дало змогу оцінити, як компанія могла б використовувати дану платформу для проведення прямих ефірів, вебінарів та внутрішніх корпоративних заходів, таких як презентації нових продуктів чи внутрішні тренінги для співробітників.

Важливим аспектом дослідження стала оцінка вимог до потокової передачі медіа в контексті бізнес-потреб компанії. Розглядалися такі параметри, як необхідна пропускна здатність мережі, затримки під час трансляцій та стійкість до втрат пакетів. У рамках цієї роботи також було розроблено рекомендації щодо оптимального вибору протоколів для різних типів подій, зокрема використання WebRTC для швидких живих подій, RTMP для матеріалів з високою роздільною здатністю, а також автоматизація вибору протоколу DASH при звичайних

мережевих умовах та WebRTC при нестабільному з’єднанні у клієнтів. Це дозволяє оптимізувати процеси передачі даних, забезпечуючи високу якість трансляції та адаптацію до змінних мережевих умов і вимог до контенту, що робить систему більш ефективною та надійною для кінцевих користувачів.

Одним із головних протоколів для живих трансляцій є RTMP, який забезпечує низьку затримку та є популярним у прямих ефірах і ігрових стрімах. З іншого боку, DASH (Dynamic Adaptive Streaming over HTTP) дозволяє підлаштовувати якість потоку під змінну пропускну здатність мережі, що мінімізує буферизацію під час перегляду контенту з високою роздільною здатністю. DASH широко використовується на платформах, таких як Netflix, YouTube для забезпечення безперервного потокового передавання за різних умов мережі [7].

Одним із практичних інструментів, що підтримує сучасні стрімінгові технології, є Ant Media Server. Він поєднує протоколи RTMP, DASH та WebRTC і надає можливість створювати адаптивні трансляції з низькою затримкою. Досвід використання Ant Media Server у різних компаніях демонструє його гнучкість та ефективність для проведення прямих ефірів, вебінарів та корпоративних презентацій.

Аналіз літератури показав, що використання адаптивних протоколів у поєднанні з машинним навчанням є перспективним напрямком для покращення якості та стабільності потокової передачі. Зокрема, інтеграція Ant Media Server відкриває нові можливості для гнучкого керування трансляціями та автоматизації вибору протоколу в реальному часі. Це дозволяє підвищити ефективність мультимедійних сервісів і забезпечити стабільну роботу навіть за складних умов мережі. Завдяки такій гнучкості ця платформа стає універсальним рішенням для різноманітних потреб: від простих прямих ефірів до масштабних стрімінгових сервісів, що вимагають адаптивної якості відео. Застосування Ant Media Server дозволяє дослідити реальні умови роботи цих протоколів, оцінити їх взаємодію та ефективність у різних сценаріях передачі даних, забезпечуючи адаптацію до змінних умов мережі та пристроїв користувачів.

Однією з ключових переваг Ant Media Server є низька затримка під час живих трансляцій. Це особливо важливо для подій, які відбуваються в реальному часі, таких як спортивні матчі чи онлайн-ігри, де навіть кількасекундні затримки можуть знизити якість сприйняття або вплинути на інтерактивність.

Іншою важливою особливістю є адаптивний стрімінг. Завдяки йому платформа автоматично підлаштовує якість відео до доступної пропускної здатності мережі користувача, мінімізуючи буферизацію та забезпечуючи плавний перегляд навіть за умов нестабільного інтернет-з'єднання.

Ant Media Server також відзначається масштабованістю, що дозволяє обслуговувати велику кількість глядачів. Завдяки підтримці кластеризації система легко адаптується до збільшення аудиторії, забезпечуючи стабільну роботу без збоїв.

Ще одна перевага цієї платформи — гнучкість у виборі протоколів. Ant Media Server може автоматично адаптуватися до різних умов трансляції, підтримуючи широкий спектр протоколів, таких як RTMP, WebRTC, HLS та DASH. Це дозволяє вибирати оптимальний протокол залежно від специфікацій користувача, типу події та мережевих умов, що забезпечує високу якість відео- та аудіопотоків навіть при змінних умовах мережі.

Завдяки всім цим можливостям Ant Media Server стає незамінним інструментом як для стрімерів і геймерів, так і для великих корпорацій та провайдерів контенту. Вона дозволяє організувати надійні, якісні трансляції, які будуть стабільними навіть у випадку коливань мережевих показників.

1.2 Методи машинного навчання

Важливим завданням системи стрімінгового відео є вибір оптимального протоколу для передачі контенту, який забезпечує ефективне використання мережевих та серверних ресурсів при збереженні високої якості досвіду користувача. Це завдання можна формалізувати як проблему багатокритерійної оптимізації, яка враховує різні параметри, такі як мережеві умови (пропускна

здатність, затримка, втрати пакетів), характеристики пристроїв користувачів та вимоги до якості трансляцій. Використання машинного навчання для прогнозування і вибору оптимального протоколу дозволяє автоматично адаптуватися до змінних умов мережі та забезпечити оптимальний баланс між швидкістю передачі та якістю контенту в реальному часі [6].

Багато сучасних досліджень зосереджуються на застосуванні машинного навчання для прогнозування параметрів мережі та вибору оптимального протоколу. Такі алгоритми використовуються для аналізу затримок, втрат пакетів та пропускної здатності в реальному часі. LSTM-моделі виявляються особливо корисними для роботи з часовими рядами та передбачення змін у стані мережі. Наприклад, дослідження показують, що нейронні мережі можуть автоматично підібрати оптимальний протокол залежно від поточних умов мережі, забезпечуючи таким чином стабільність та якість потокової передачі [7].

1.3 Постановка задачі дослідження

Розвиток мультимедійних технологій і популярність онлайн-стрімінгу вимагають вибору ефективних протоколів для передавання аудіо- та відеоконтенту. У сучасних умовах популярними є протоколи RTMP (Real-Time Messaging Protocol) та DASH (Dynamic Adaptive Streaming over HTTP), які забезпечують якісне передавання даних. Однак питання автоматизованого вибору відповідного протоколу в різних умовах залишається актуальним завданням, що потребує застосування методів машинного навчання.

Основна проблема полягає у тому, що під час потокової передачі мережеві умови можуть змінюватися в реальному часі (наприклад, зниження пропускної здатності або збільшення затримки). Використання лише одного протоколу може призвести до погіршення якості трансляції або виникнення затримок. Тому необхідний динамічний вибір протоколу передачі в залежності від параметрів мережі та типу контенту. Застосування машинного навчання дає змогу створити автоматизоване рішення, що передбачатиме та відповідатиме на зміни в мережі.

Основною метою кваліфікаційної роботи є розробка методу вибору оптимального стрімінгового протоколу в умовах змінних мережевих параметрів з використанням методів машинного навчання. Для реалізації цієї мети необхідно розглянути кілька ключових аспектів.

По-перше, необхідно провести аналіз області застосування, щоб зрозуміти важливість автоматизації процесу вибору протоколу для стрімінгових сервісів. Цей аналіз допоможе виокремити проблеми та складнощі, що виникають під час вибору оптимального протоколу в реальному часі, враховуючи умови мережі, такі як пропускна здатність, затримка та втрати пакетів.

По-друге, потрібно провести комплексний аналіз методів машинного навчання, які можуть бути застосовані для прогнозування та вибору протоколів. Однією з найбільш підходящих моделей є Long Short-Term Memory (LSTM), яка є спеціалізованою архітектурою нейронних мереж для роботи з часовими рядами і дозволяє прогнозувати зміни в мережевих параметрах, таких як пропускна здатність і затримка. Сильні сторони та обмеження цієї моделі будуть оцінені для визначення її придатності для вибору протоколу у стрімінгових сервісах [8].

По-третє, необхідно розробити метод автоматизованого вибору оптимального протоколу для відео- та аудіотрансляцій на основі машинного навчання, що враховує мережеві умови, характеристики пристроїв користувачів та вимоги до якості трансляцій [9].

Завдяки цьому методу планується забезпечити оптимальний баланс між швидкістю передачі та якістю контенту, що дозволить покращити ефективність потокового передавання даних.

Висновки до розділу 1

1. Ant Media Server забезпечує високу ефективність у передаванні відео та аудіо з низькою затримкою, що робить його ідеальним вибором для реального часу трансляцій. Завдяки підтримці таких протоколів, як RTMP та DASH, ця платформа здатна забезпечити стабільний і якісний потік навіть у складних

мережевих умовах. Це робить її зручним інструментом для тестування різних сценаріїв та оптимізації потокової передачі з урахуванням змінних параметрів мережі.

2. Сучасні технології машинного навчання відкривають нові можливості, дозволяючи значно підвищити точність прогнозів завдяки здатності працювати з великими обсягами даних. За допомогою таких методів, як глибинне навчання та нейронні мережі, можливо виявляти складні патерни та взаємозв'язки в історичних даних ринку, що забезпечує точніші прогнози та більш ефективне прийняття рішень. Алгоритми машинного навчання здатні автоматично адаптуватися до змінюваних умов ринку, дозволяючи фінансовим аналітикам та інвесторам знижувати ризики і покращувати стратегії управління активами.

3. Модель LSTM є одним з найбільш підходящих інструментів для прогнозування та аналізу часових рядів у контексті змінних мережевих умов. Вона дозволяє ефективно передбачати варіації таких параметрів, як пропускна здатність та затримка, що є критичними для вибору оптимального протоколу в реальному часі. LSTM допомагає зберігати важливу інформацію з попередніх проміжків часу, що робить її особливо корисною для прогнозування та оптимізації процесів потокової передачі.

4. Інтеграція технологій машинного навчання та автоматизації створює потужні можливості для розробки гнучких рішень, які здатні ефективно враховувати складність та мінливість ринкових умов. Завдяки здатності машинного навчання до самонавчання та адаптації, ці системи можуть оперативно реагувати на зміни в економічному середовищі, що дозволяє не лише точніше прогнозувати майбутні тренди, але й автоматично коригувати стратегії в реальному часі. Такий підхід значно підвищує ефективність управління фінансовими активами, оптимізуючи процеси прийняття рішень та знижуючи ризики, що виникають через непередбачуваність ринку.

2 РЕЗУЛЬТАТИ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ У МЕЖАХ НАУКОВОЇ ЗАДАЧІ

2.1 Проведення аналізу методів та технологій

Під час потокової передачі ключовими факторами є коротка затримка запуску, плавне відтворення та висока швидкість передачі даних. Традиційні технології для цього використовували такі протоколи, як RTSP, RTMP та MS-RTSP. Ці протоколи встановлюють сеанс між клієнтом і сервером, який підтримується до завершення з'єднання. Протягом сеансу користувач може керувати передаванням, виконуючи дії, такі як відтворення, пауза, запис та завершення сесії.

Власні протоколи потокової передачі на основі сеансу широко використовуються для аудіоконференцій та відеосеансів із низькою затримкою, що забезпечує швидкий запуск і хорошу взаємодію з користувачем. Однак такі підходи мають певні недоліки:

1. Необхідність спеціалізованих серверів. Для налаштування та підтримки таких серверів потрібні професійні навички, що робить їх використання дорогим у великомасштабних розгортаннях.
2. Трафік UDP. Більшість таких протоколів використовують UDP, який часто блокується брандмауерами та NAT за замовчуванням, ускладнюючи їх використання.
3. Витрати ресурсів сервера. Сервер повинен відстежувати стан кожного сеансу, що обмежує масштабованість та вимагає значних обчислювальних ресурсів.
4. Фіксований бітрейт. Бітрейт, за яким сервер передає вміст, збігається з бітрейтом кодування медіафайлів та відтворення на клієнті. Це дозволяє підтримувати стабільний рівень буфера та ефективно використовувати мережеві ресурси.

Однак у випадках значної затримки або втрат пакетів, швидкість наповнення буфера клієнта може впасти нижче швидкості споживання. Це може

призвести до виснаження буфера, що спричинить паузу у відтворенні, погіршуючи користувацький досвід.

Іншим до потокової передачі є поступове (прогресивне) завантаження HTTP, що використовує звичайний веб-сервер замість потокового сервера для передачі відео. Відео завантажується як один великий файл, і клієнт може почати відтворення після завантаження перших кількох секунд. Сайти, такі як YouTube і Vimeo, використовують цей підхід. Перевагою такого підходу є використання TCP, що спрощує роботу з брандмауерами та NAT і можливість доставки пакетів великими порціями [11].

Недоліком є те, що не можна адаптувати якість відео до змін у мережі — всі клієнти отримують однаковий бітрейт. Також, неефективне використання смуги пропускання: уповільнення завантаження запобігає зайвим затримкам, але не вирішує проблему повністю.

Ще однією технологією є адаптивна потокова передача HTTP дозволяє користувачам отримувати доступ до відеоконтенту на різних пристроях – від телефонів і ноутбуків до смарт-телевізорів. Технологія визначає пропускну здатність мережі та продуктивність процесора в режимі реального часу, динамічно регулюючи якість відео для забезпечення плавного перегляду.

Спочатку цю технологію запропонувала компанія Move Networks у 2006 році, а Microsoft інтегрувала її в IIS 7.0 у 2008 році під час трансляції Олімпійських ігор. Пізніше такі компанії, як Apple, Netflix та Akamai, активно впровадили цей підхід.

Особливістю та перевагою є гібридний підхід, що поєднує елементи потокової передачі та прогресивного завантаження через HTTP. Також підтримка брандмауера, він працює без потреби в NAT, що спрощує налаштування мереж. Має можливість масштабування, що зменшує витрати на сервери завдяки використанню CDN та оптимізації HTTP [12].

Недоліками є витрати на кодування та зберігання, оскільки кодування відео у кількох бітрейтах збільшує витрати. Також складність реалізації для підтримання глобальної якості вимагає додаткових ресурсів.

Ця технологія є ефективним рішенням, хоча її впровадження складніше порівняно з традиційними методами потокового передавання. Проте переваги масштабованості та сумісності з різними мережами роблять її одним із провідних стандартів у сучасних відеосервісах.

Результат аналізу технологій представлено в таблиці 2.1, де показано, що прогресивне завантаження є найпростішим варіантом, але має обмежену функціональність. Потокове передавання забезпечує доступ до будь-якої частини відео, але вимагає складнішої конфігурації мережі. Адаптивне потокове передавання дозволяє змінювати якість відео в реальному часі, що робить його оптимальним для сучасних платформ із динамічними мережевими умовами.

Таблиця 2.1 – Порівняння доступних технологій

Особливість	**Прогресивне завантаження**	**Потокове передавання**	**Адаптивне потокове передавання**
Відтворення відео на шкалі часу	Лінійний характер, неможливо перемотати вперед до незавантаженого вмісту	Можлива навігація вперед	Можливе перемикання між сегментами для відтворення на будь-якій точці
Транспортний протокол	HTTP через TCP (порт 80)	RTMP, RTSP через TCP/UDP	HTTP через TCP (порт 80)
Попередня обробка	Не потребує спеціальної обробки	Сервер керує передаванням фрагментів	Потрібне кодування для кількох бітрейтів і пошук у маніфесті
Моніторинг та контроль	Відсутній моніторинг чи контроль	Підтримка стану та серверний контроль	Ефективний контроль, клієнт перемикає потоки відповідно до умов мережі
Зберігання медіа	Файл зберігається в кеші	Отримує фрагменти й видаляє їх після відтворення	Запити на фрагменти з можливістю кешування
Ключові переваги	Легка настройка, немає ліцензійних вимог	Можливість доступу до будь-якої частини відео без очікування	Гнучка якість потоку залежно від мережі
Недоліки	Витрати пропускної здатності через незавантажений контент	Вимагає спеціальної мережевої конфігурації	Обмежена підтримка стандартів на деяких платформах
Приклад платформ	YouTube, Vimeo	Hulu	Netflix, BBC iPlayer
Безпека вмісту	Вміст зберігається в кеші, менш безпечний	Без кешування, але є ризик захоплення потоку	Інтеграція DRM для захисту контенту

Продовження таблиці 2.1

Особливість	Прогресивне завантаження	Потокове передавання	Адаптивне потокове передавання
Вимоги до клієнта/сервера	Будь-який HTTP-сервер (наприклад, Apache)	Потоковий сервер (Wowza, Red5)	HTTP-сервер із підтримкою HLS чи DASH
Поведінка на повільному з'єднанні	Довге очікування перед відтворенням	Відтворення з затримками або наданими варіантами якості	Перемикання на низьку якість для підтримки відтворення
Основне визначення	Файл передається цілком через HTTP	Сервер надсилає фрагменти за запитом клієнта	Клієнт обирає фрагменти через маніфест
Використання пропускної здатності	Менш ефективно, оскільки весь файл завантажується відразу	Більш ефективне, завантажуються лише потрібні фрагменти	Кешування фрагментів у CDN для збереження трафіку

Приклад порівняння продуктивності стрімінгових протоколів [12] представлений на рисунку 2.1.

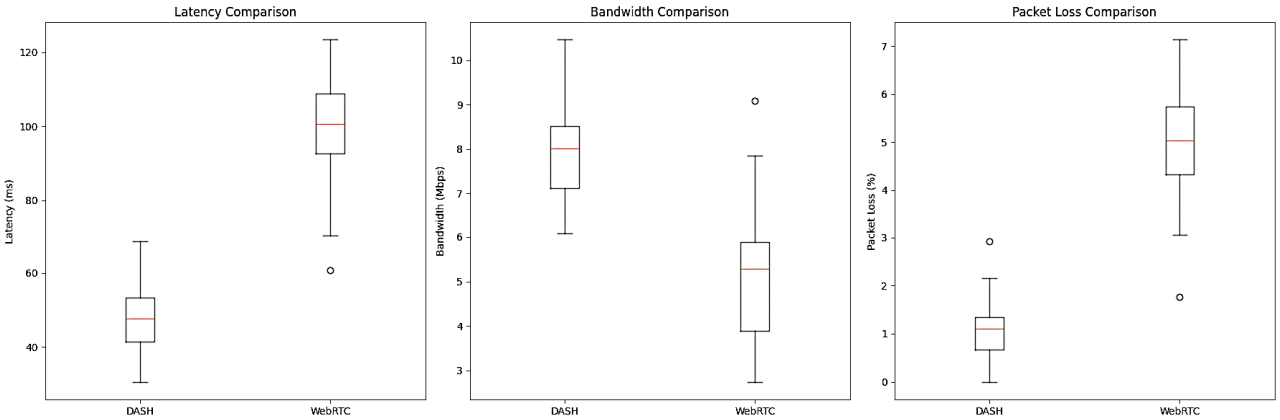


Рисунок 2.1 – Графіки результатів порівняння продуктивності стрімінгових протоколів.

2.2 Дослідження реалізацій потокової передачі

RTMP (Real-Time Messaging Protocol) — це протокол для передачі аудіо, відео та даних в реальному часі через мережу Інтернет. Спочатку розроблений компанією RTMP використовувався для передачі медіа-контенту в додатках Flash. Попри зникнення Flash, RTMP залишається популярним для потокових

трансляцій, зокрема у сфері онлайн-відео, живих подій, вебінарів, та відеоігор. RTMP працює за принципом клієнт-сервер, де сервер приймає відео та аудіо дані від джерела, а потім передає їх клієнтам або глядачам, які підключаються до сервера для перегляду трансляції.

Протокол забезпечує стабільний потік даних і надає можливість адаптації під різні мережеві умови, хоча і не має вбудованих механізмів для адаптивного бітрейту (це властиво протоколам типу HLS або DASH).

Попри свої переваги, RTMP має обмеження в підтримці сучасних мобільних пристроїв та не є оптимальним для доставки контенту через HTML5 браузери без додаткових плагінів. Тому сучасні платформи часто комбінують RTMP для прийому потоків та інші протоколи (наприклад, HLS або WebRTC) для доставки відео до кінцевих користувачів [10].

Smooth Streaming – це технологія адаптивної потокової передачі за допомогою HTTP через Silverlight, що дозволяє динамічно підлаштовувати якість відео під умови мережі та пропускну здатність користувача (рисунок 2.2). Вона базується на медіасервісах IIS і підтримує високу якість перегляду, яка масштабується за допомогою мереж CDN (мереж розповсюдження контенту).

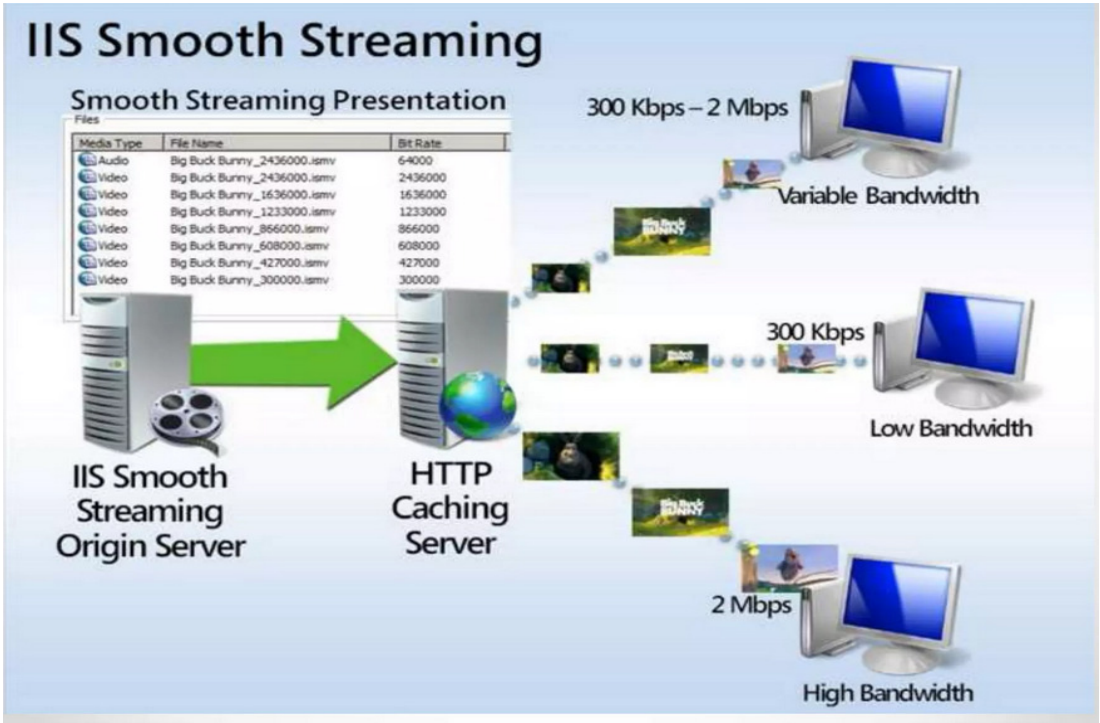


Рисунок 2.2 – Схема архітектури порівняння продуктивності Smooth Streaming.

Основні можливості:

- Автоматичне регулювання якості відео для плавного відтворення на різних пристроях.
- Підтримка функцій, як-от пауза, перемотування назад та зміна ракурсів камери.
- Кешування фрагментів відео для зменшення затримок і підвищення швидкості доставки.

Робочий процес :

1. Придбання контенту. Вміст високої якості отримується разом із субтитрами, аудіо доріжками та додатковими матеріалами.
2. Кодування. Відео розбивається на фрагменти MP4, що легко кешуються та масштабуються. Маніфест файлів містить метадані про кодеки та бітрейти.
3. Приблизне редагування. Використовується для швидкого створення оглядів або ключових моментів під час трансляцій у прямому ефірі. Цей процес мінімізує витрати та пришвидшує публікацію контенту.
4. Доставка. Контент доставляється через Інформаційні служби Інтернету, що підтримують динамічний і персоналізований розподіл.

Плавне потокове передавання (на вимогу): Використовує HTTP з кешуванням для доставки подій у прямому ефірі. Додає функції, такі як пауза, повторне відтворення та вихід у прямому ефірі. Функції та особливості Smooth Streaming:

1. Плавне потокове передавання (на вимогу). Використовує HTTP з кешуванням для доставки подій у прямому ефірі. Додає функції, такі як пауза, повторне відтворення та вихід у прямому ефірі.
2. Регулювання швидкості потоку. Автоматично визначає формат і буфер кодування, підтримуючи різні аудіо- та відеоформати.
3. Веб-списки відтворення. Запобігають несанкціонованому відтворенню третіми сторонами, дозволяючи гнучку інтеграцію клієнтських і серверних списків.

4. Розширене журналювання. Забезпечує аналіз і моніторинг даних для великих мереж.
5. Масштабування продуктивності. Використовує CDN (наприклад, Akamai) для підвищення надійності та ефективності кешування.
6. Споживання. Забезпечує однакову якість послуг на різних платформах і пристроях, таких як смарт-телевізори та приставки.

Протокол HLS (HTTP Live Streaming) дозволяє надсилати аудіо та відео з веб-сервера для відтворення на різних пристроях, таких як iPhone, iPad, iPod touch, Apple TV та настільні комп'ютери. Він підтримує як прямі трансляції, так і відео на вимогу (VOD), адаптуючи якість потоку в реальному часі залежно від пропускної здатності мережі. HLS також забезпечує шифрування та автентифікацію для захисту даних і конфіденційності користувачів. Компоненти архітектури HLS:

1. Серверний компонент:
   - Медіакодер приймає вхідні дані, кодує їх у формати H.264 (відео) та AAC (аудіо) й упаковує як транспортний потік MPEG-2.
   - Сегментатор потоку розбиває відео на невеликі файли .ts однакової тривалості й створює індексний файл .M3U8 із посиланнями на ці сегменти. Він також шифрує медіасегменти й створює ключі для дешифрування.
   - Сегментатор файлів використовується для поділу готових файлів на сегменти й забезпечує доставку VOD.
2. Компонент розподілу:
   - Використовує веб-сервери на базі HTTP для доставки сегментів. Додаткові серверні модулі не потрібні, а CDN допомагають масштабувати доставку для великих аудиторій.
3. Клієнтський компонент:
   - Клієнтське програмне забезпечення отримує файл індексу через URL-адресу, яка вказує на потік. Індексний файл містить розташування медіасегментів, ключі для дешифрування та

інформацію про доступні потоки. Після завантаження достатньої кількості даних здійснюється безперервне відтворення.

2.3 Дослідження типів сеансів

Існує два типи сеансів пряма трансляція та відео на вимогу. Нижче наведений їх ширший опис .

1. Пряма трансляція (Live), представлена на рисунку 2.3, відбувається у режимі реального часу з поступовим завантаженням сегментів.
2. Відео на вимогу (VOD) полягає в наданні доступу клієнту до повного переліку медіафайлів із можливістю навігації між сегментами за індексним файлом.

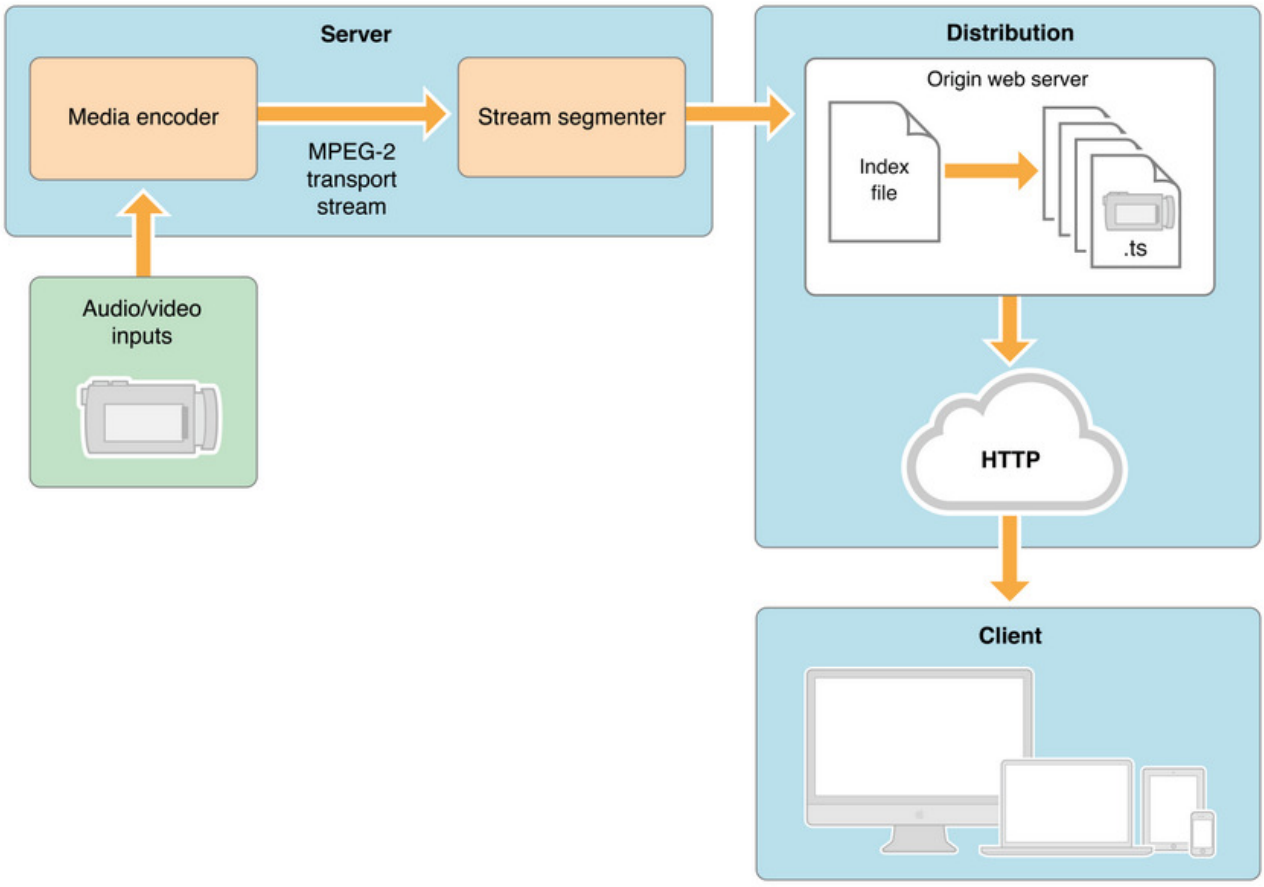


Рисунок 2.3 – Схема архітектури прямої трансляції робочого процесу HTTP.

Опис компонентів представлених на рисунку 2.3:

1. Серверний компонент
   - Медіакодер приймає вхідні дані та кодує їх у відеоформат H.264 і аудіоформат AAC, забезпечуючи сумісність із пристроями користувача. Кодування в транспортний потік MPEG-2 дозволяє інкапсулювати контент для передачі.
   - Сегментатор потоку розбиває транспортний потік на рівні за тривалістю сегменти у форматі .ts. Він також генерує файл індексу .M3U8, який містить посилання на сегменти для безперервного відтворення. Сегменти шифруються з використанням ключів для забезпечення безпеки.
   - Сегментатор файлів використовується для розподілу вже наявних файлів на сегменти однакової довжини, що дозволяє повторно використовувати аудіо- та відеоконтент для відео на вимогу (VOD).
2. Компонент розподілу
   - Включає веб-сервер, який використовує HTTP для доставки контенту без потреби в спеціальних модулях. Використання CDN підвищує продуктивність і масштабованість.
3. Клієнтський компонент
   - Клієнт отримує файл індексу через URL, який містить інформацію про доступні сегменти та ключі для дешифрування. Після завантаження достатньої кількості даних контент відтворюється безперервно.

Існують два типи сеансів:

1. Пряма трансляція, де потік створюється в реальному часі, забезпечуючи глядачам актуальний контент.

2. Відео на вимогу (VOD), що надає користувачам доступ до повного переліку файлів із можливістю навігації. Індексний файл містить усі сегменти з початку трансляції, що забезпечує повний контроль відтворення.

Технологія Content Delivery Network (CDN) забезпечує ефективне, безпечне та масштабоване передавання контенту на різні пристрої, підтримуючи

як трансляції в реальному часі, так і перегляд на вимогу. CDN – це мережа серверів, розподілених географічно, які працюють разом для забезпечення швидкої та ефективної доставки контенту кінцевим користувачам [9]. CDNs використовуються для оптимізації передачі різних видів контенту, таких як веб-сторінки, зображення, відео, аудіо та програмне забезпечення, шляхом кешування даних на серверах, які знаходяться ближче до кінцевих користувачів.

Основні принципи роботи CDN:

- CDN зберігає копії статичного контенту (наприклад, HTML-файлів, зображень, відео) на декількох серверах, які знаходяться в різних географічних точках. Коли користувач запитує ресурс, CDN перенаправляє запит до найближчого сервера, що дозволяє зменшити час завантаження і знизити навантаження на основні сервери.
- CDN дозволяє розподіляти трафік між кількома серверами, що допомагає уникнути перевантаження окремих вузлів мережі, збільшуючи доступність і швидкість доставки контенту.
- Завдяки географічно розподіленим серверам, CDN забезпечує мінімізацію затримок при доставці контенту. Наприклад, відео або аудіо контент, що передається через CDN, завжди обробляється сервером, найближчим до користувача, що знижує час завантаження.
- CDN дозволяє ефективно обробляти великий обсяг запитів, особливо під час пікових навантажень, таких як великі онлайн-трансляції або розповсюдження популярного контенту. Вона забезпечує надійність і постійну доступність сервісів.

Основні переваги використання CDN:

- Швидкість доставки контенту. Завдяки кешуванню і зменшенню затримок, користувачі отримують контент значно швидше.
- Зниження навантаження на основні сервери. CDN зменшує кількість запитів до основного сервера, що підвищує стабільність роботи вебсайту або онлайн-сервісу.

- Підвищена безпека. CDN може забезпечити додатковий рівень безпеки через захист від DDoS-атак, оскільки зловмисники мають більше труднощів у націлюванні на сервери, які географічно розподілені.
- Масштабованість. CDN дозволяє легко масштабувати інфраструктуру для обслуговування більшої кількості користувачів, без необхідності істотних інвестицій у додаткові фізичні сервери.

У стрімінгових сервісах, таких як Netflix, YouTube, або Twitch, CDN є ключовим елементом для ефективної доставки контенту в реальному часі. Вони дозволяють забезпечити високу якість відео, мінімізувати буферизацію та оптимізувати пропускну здатність для користувачів по всьому світу. Завдяки технології адаптивного стрімінгу (наприклад, DASH або HLS), CDN можуть динамічно регулювати якість відео в залежності від швидкості інтернет-з'єднання користувача.

У випадку високих навантажень, таких як популярні прямі трансляції або глобальні події, CDN допомагають масштабувати ресурси, щоб справитися з величезним трафіком і уникнути перевантажень на окремих серверах. Таким чином, CDN є основою для сучасних інфраструктур потокових сервісів, допомагаючи покращити доступність, швидкість і стабільність доставки контенту.

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) – це міжнародний стандарт адаптивної потокової передачі, розроблений у 2010 році та затверджений ISO у 2012 році. Він використовує інфраструктуру HTTP, що дозволяє передавати контент на різні пристрої: смарт-телевізори, комп’ютери, планшети та смартфони [12, 13].

Основні особливості:

- Маніфест у форматі XML описує доступний контент, що може містити різні аудіодоріжки, субтитри та відео з кількома ракурсами. Це дозволяє легко інтегрувати рекламу та створювати динамічний досвід.
- MPEG-DASH може змінювати якість відео залежно від стану мережі та продуктивності пристрою.

– Незважаючи на схожість з іншими протоколами, такими як HLS або Smooth Streaming, MPEG-DASH забезпечує кращу інтеграцію на різних платформах і пристроях.

Компанії та підтримка:

– Google, Samsung, Netflix та інші лідери індустрії активно використовують MPEG-DASH для доставки адаптивного контенту. Ця технологія дозволяє забезпечити високу якість перегляду навіть за умов змінної пропускної здатності.

На рисунку 2.4 представлено модель даних Media Presentation Description (MPD), яка використовується для організації та опису мультимедійного контенту в адаптивному стрімінгу. Ця модель дозволяє деталізувати структуру контенту, забезпечуючи його сегментацію, вибір окремих компонентів і адаптацію до мережевих умов.

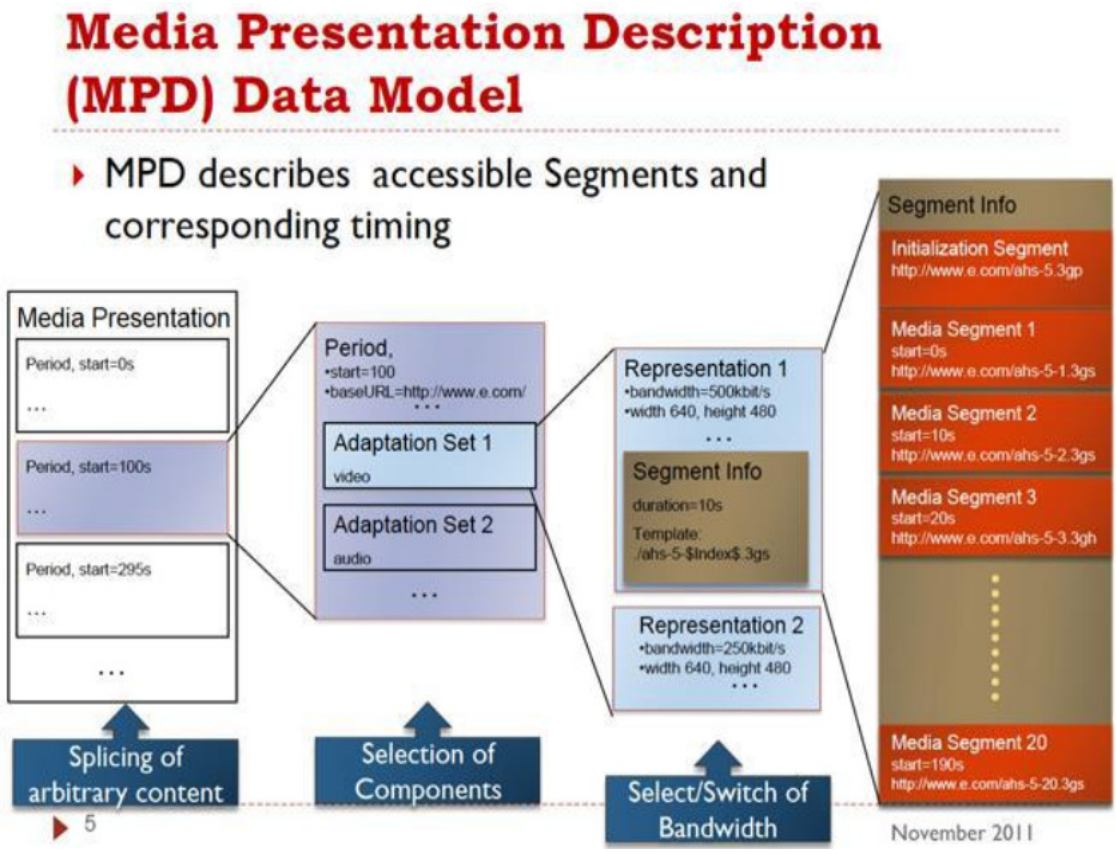


Рисунок 2.4 – Модель MPD у MPEG-DASH.

Медіапрезентація складається з періодів, кожен із яких представляє логічну частину контенту з визначеним часовим інтервалом. Наприклад, один період може охоплювати початкову частину відео, а інший — наступний етап. Кожен період містить набори адаптації, які відповідають за конкретні компоненти медіаконтенту, такі як відео, аудіо чи субтитри. У межах цих наборів визначаються різні варіанти представлення контенту, наприклад, відео з різними роздільними здатностями або рівнями бітрейту. Це дозволяє забезпечити відтворення контенту з урахуванням особливостей пристрою користувача та умов мережі [14-16].

Ключовим елементом MPD є сегментація контенту, де кожне представлення розбивається на невеликі частини — сегменти. Ці сегменти, тривалістю кілька секунд, можуть бути незалежно завантажені й відтворені, що забезпечує гнучкість і адаптивність під час потокового передавання. Початковий сегмент містить необхідну інформацію для початку відтворення, а наступні — основний медіаконтент.

Модель MPD також враховує можливість перемикання між різними рівнями якості під час відтворення. Це означає, що при зміні пропускної здатності мережі система може автоматично перейти на інше представлення, наприклад, з високої роздільної здатності на нижчу, щоб уникнути буферизації. Така динамічність дозволяє забезпечити плавність і стабільність роботи навіть у складних мережевих умовах.

Загалом, MPD є важливим компонентом сучасного адаптивного стрімінгу, оскільки вона забезпечує ефективну організацію контенту, гнучкість у його відтворенні та оптимізацію під різні умови використання. Завдяки своїм можливостям ця модель використовується у таких платформах, як Netflix, YouTube, а також у відеоконференціях та інших стрімінгових сервісах [17-19].

MPEG-DASH – це гнучка технологія потокової передачі з підтримкою прямого ефіру, відео на вимогу (VOD) та функцій зсуву часу (наприклад, пауза та перемотування). Нижче наведено ключові особливості цієї технології:

1. Інтеграція з існуючими мережами. Підтримка CDN, NAT, проксі та брандмауерів, що забезпечує стабільну роботу.

2. Запити діапазону байтів. Сегменти зберігаються безперервно, а клієнти запитують лише необхідні діапазони.

3. Клієнтський контроль. Простий сервер може розміщувати файли, але якість залежить від поведінки клієнта.

4. Перемикання треків. Легке перемикання між субтитрами та аудіо.

5. Режим Trick Play: Функції паузи, перемотування та повторного відтворення завдяки ключовим кадрам.

6. Загальне шифрування. Використання AES-128 для одноразового шифрування з підтримкою різних DRM-схем.

7. Профілі для різних сценаріїв. Netflix розробив профіль для VOD, а HLS – для прямого ефіру.

8. Ефективна адресація сегментів: Схеми для спрощеного доступу до контенту, що полегшує створення маніфестів.

MPEG-DASH забезпечує високу гнучкість і адаптивність, дозволяючи легко інтегрувати різний контент та доставляти його на широкий спектр пристроїв.

Висновки до розділу 2

1. Проведений аналіз сучасних методів і технологій потокової передачі показав, що ключовими факторами для забезпечення якісного стрімінгу є коротка затримка запуску, плавне відтворення та адаптація до мережевих умов. Найбільш перспективними є адаптивні технології, такі як MPEG-DASH і HLS, які динамічно змінюють якість потоку в залежності від доступної пропускної здатності мережі.

2. Традиційні протоколи, такі як RTSP і RTMP, хоча і забезпечують низьку затримку та стабільність передачі, мають обмеження у масштабованості та підтримці сучасних пристроїв. Крім того, їхня залежність від спеціалізованих

серверів та проблеми з проходженням через NAT і брандмауери ускладнюють їх використання в глобальних масштабах.

3. Прогресивне завантаження HTTP показало свою простоту реалізації та сумісність із різними пристроями. Однак обмежена функціональність, зокрема неможливість адаптувати якість відео до змін у мережі, робить цей підхід менш привабливим для сучасних динамічних платформ.

4. Адаптивна потокова передача через HTTP (HLS, MPEG-DASH) виділяється своєю масштабованістю та гнучкістю. Використання CDN, підтримка шифрування та функції динамічного регулювання якості забезпечують оптимальний користувацький досвід навіть за складних мережевих умов.

5. Аналіз реалізацій таких протоколів, як RTMP, Smooth Streaming, HLS і MPEG-DASH, підтвердив їхню актуальність у різних сценаріях. Зокрема, RTMP залишається популярним для прийому трансляцій, тоді як HLS і MPEG-DASH демонструють високу ефективність у доставці контенту кінцевим користувачам.

6. Використання CDN є важливим елементом у сучасних потокових платформах. Завдяки розподілу навантаження та мінімізації затримок, CDN забезпечує швидку та стабільну доставку контенту навіть при високих навантаженнях, таких як глобальні трансляції або популярні події.

7. MPEG-DASH підтвердив свою універсальність та ефективність завдяки інтеграції з існуючими мережами та підтримці функцій адаптивної передачі. Можливість зміни якості потоку, інтеграція DRM і підтримка різних профілів роблять його провідним стандартом у сфері потокової передачі.

8. Загальний висновок аналізу полягає в тому, що вибір технології повинен враховувати специфічні вимоги платформи, такі як підтримка пристроїв, масштабованість і адаптація до мережевих умов. Адаптивні протоколи, такі як MPEG-DASH, представляють оптимальне рішення для забезпечення якісного і гнучкого стрімінгу в сучасних умовах.

3 РЕАЛІЗАЦІЯ МЕТОДУ ВИБОРУ АПАРАТНИХ РЕСУРСІВ

3.1 Опис архітектури системи

Пряма (Live) трансляція полягає в процесі передачі аудіо та відео сигналів від джерела на сервер, де вони обробляються та передаються кінцевим користувачам через відповідні протоколи, забезпечуючи можливість перегляду відео в реальному часі як зображено на рисунку 3.1.

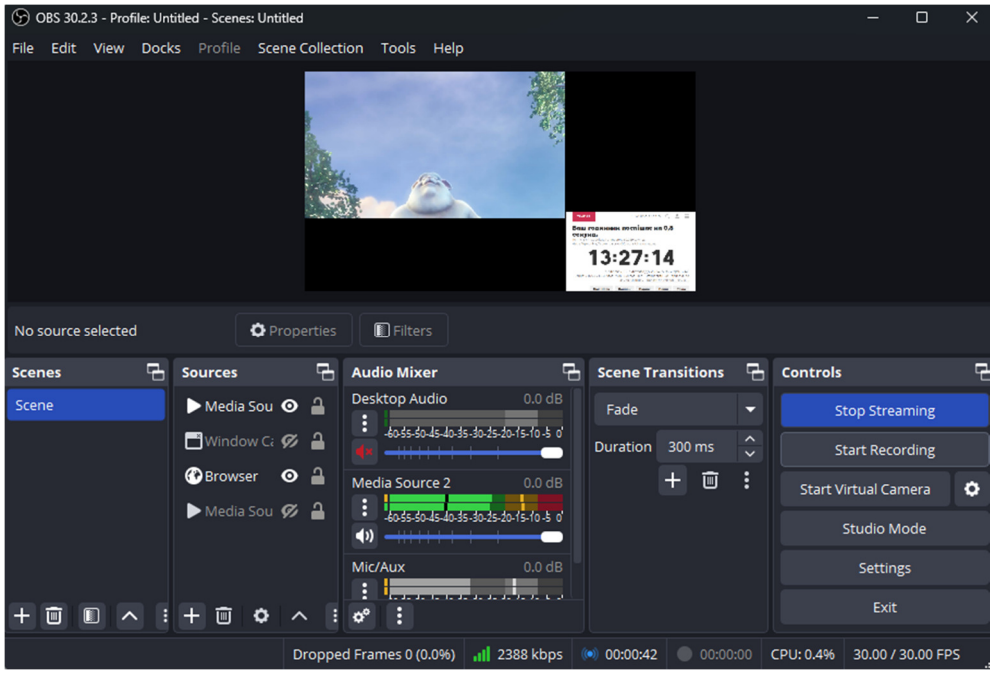


Рисунок 3.1 –Потокова передача аудіо та відео на сервер.

Передача аудіо та відео на сервер здійснюється за допомогою протоколів RTMP та WebRTC. RTMP в Ant Media Server краще за WebRTC, коли потрібно забезпечити стабільну трансляцію для великої аудиторії з високою якістю відео, тоді як WebRTC є більш підходить для двостороннього зв'язку та взаємодії в реальному часі [20, 21].

Для кінцевих користувачів важливо враховувати, що підключення можуть бути різними, а пристрої — дуже різноманітними. У таких умовах автоматичний вибір протоколу між WebRTC або DASH може бути оптимальним рішенням. На

рисунку 3.2 показано процес передачі потокового аудіо та відео разом із користувацьким плеєром WebRTC. Червоним кольором виділено час на таймері, який демонструє, що використання WebRTC забезпечує затримку у 5 секунд, що є ключовим для інтерактивного досвіду.

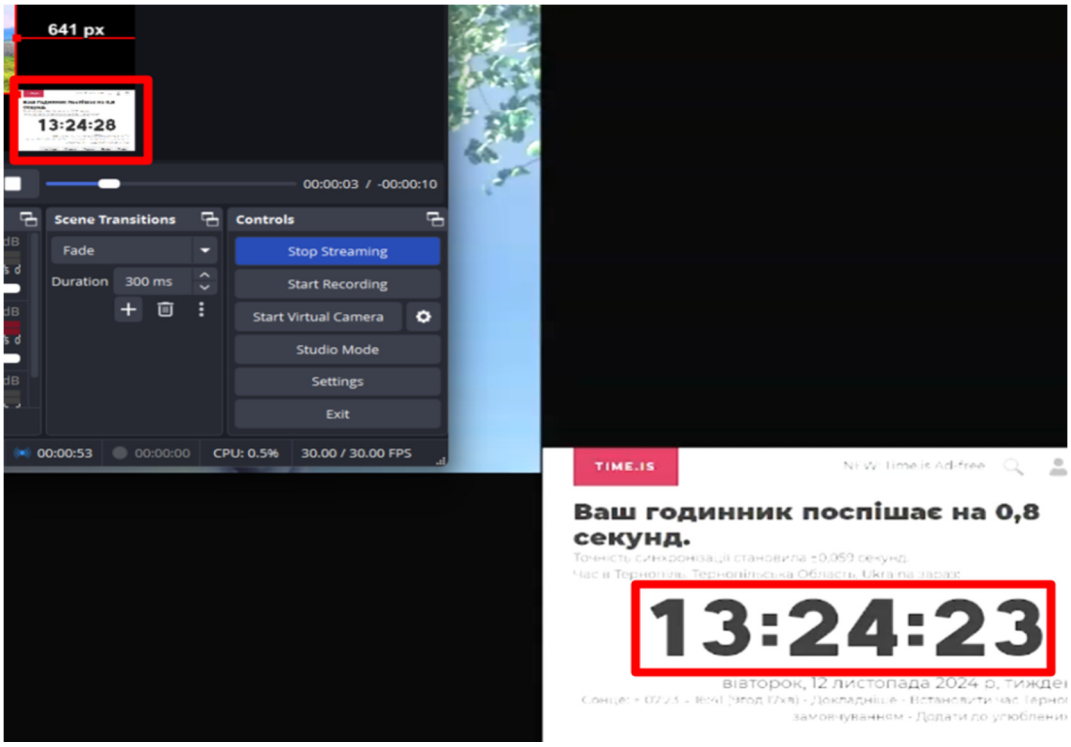


Рисунок 3.2 – Потокова передача аудіо та відео та відтворення через WebRTC.

На рисунку 3.3 проведено тест використовуючи користувацький плеєр DASH . Червоним виділено час на таймері що у випадку DASH забезпечує затримку в 9 секунд. У порівнянні з рисунком 3.2, де використовується плеєр WebRTC із затримкою у 5 секунд, протокол DASH демонструє більший час затримки, проте забезпечує стабільну передачу відео, адаптуючи якість потоку до мережевих умов. Ця різниця ілюструє переваги WebRTC для інтерактивного контенту та DASH для забезпечення безперервності відтворення.

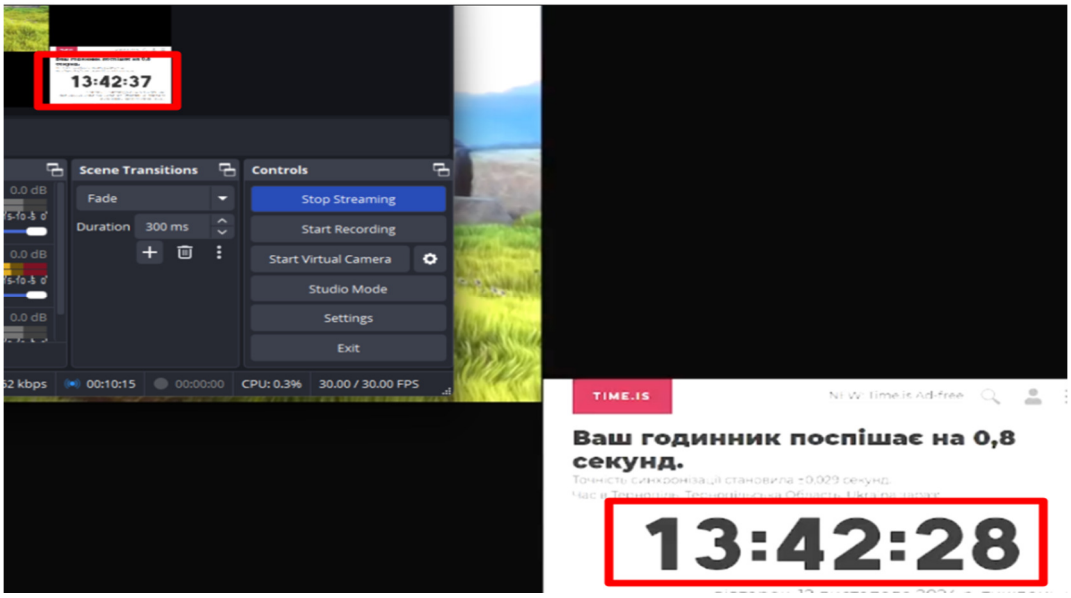


Рисунок 3.3 – Потокова передача аудіо та відео та відтворення через DASH.

Пряма трансляція через протокол HLS представлена на рисунку 3.4, характеризується затримкою у 13 секунд через процес буферизації [22, 23]. Цей показник значно перевищує затримку, яку забезпечують протоколи WebRTC та DASH, що робить їх більш ефективними для інтерактивних або адаптивних потокових сервісів.

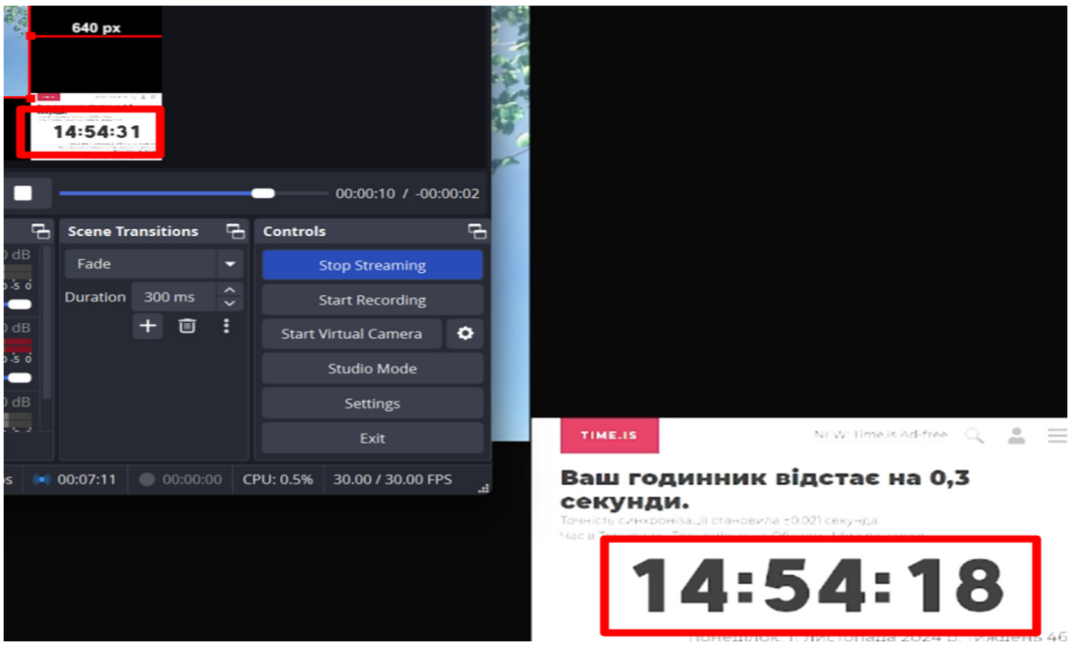


Рисунок 3.4 – Потокова передача аудіо та відео та відтворення через HLS.

У таких умовах оптимальним рішенням може бути автоматичний вибір протоколу між WebRTC та DASH. WebRTC забезпечує мінімальну затримку, що особливо важливо для інтерактивних трансляцій і відеозв'язку. Водночас DASH дозволяє адаптувати якість відео до змінних мережевих умов, забезпечуючи стабільне відтворення навіть за нестабільного з'єднання. На відміну від HLS, який через буферизацію має вищу затримку, ці протоколи значно покращують досвід користувачів, пропонуючи більш гнучкі та ефективні рішення для різних сценаріїв використання [24, 25].

Вимірювання затримки за допомогою утиліти ping — це один із найбільш ефективних способів оцінки часу, необхідного для передачі пакету даних між двома точками в мережі [26]. Команда ping надсилає ICMP-запит до вказаного хоста і вимірює час, що пройшов від відправлення запиту до отримання відповіді, що дозволяє визначити затримку в мілісекундах. Цей процес допомагає швидко оцінити якість мережевого з'єднання та виявити потенційні проблеми, такі як висока затримка або втрати пакетів. Затримка до 50 мс вважається хорошим показником для більшості застосунків, тоді як значення понад 100 мс можуть свідчити про проблеми з мережевим з'єднанням.

Цей код викликає утиліту ping з консолі для перевірки доступності заданого IP-адреси. Спочатку формуються параметри команди ping з вказівкою кількості пакетів для відправлення. Використання 10 пакетів дозволяє зібрати достатньо даних для аналізу якості з'єднання, при цьому мінімізуючи час тестування та навантаження на мережу. Після цього команда ping виконується, і результати тесту, такі як час затримки та статистика втрат пакетів, зчитуються та обробляються.

```
StringBuilder outputResult = new StringBuilder();
String pingCmd = String.format("ping -c 10 %d", ip);
try {
     Runtime runtime = Runtime.getRuntime();
     Process process = runtime.exec(pingCmd);
     InputStreamReader streamReader = new
InputStreamReader(process.getInputStream());
```

```
        BufferedReader in = new BufferedReader(streamReader);
        String inputLine;
        while ((inputLine = in.readLine()) != null) {
            outputResult.append(inputLine);
        }
        in.close();
    } catch (IOException e) {
        logger.warn("createLatency", e);
    }
```

Після виконання команди ping, консольний результат містить інформацію про затримку для кожного пакету, а також статистику в кінці тесту. Після цього результат з консолі за допомогою регулярні вирази (RegEx).

```
String AVERAGE_PATTERN = "=\\s*\\d+\\.\\d+\\/(\\d+).\\d+";
Pattern patternCompile = Pattern.compile(AVERAGE_PATTERN,
Pattern.MULTILINE);
        // Create a matcher object
        Matcher matcher = patternCompile.matcher(input);
        // Find the average value
        if(!matcher.find()) {
            logger.warn("matcher not found");
            return "0";
        }
        // Extract the captured group (the number part)
        String averageValue = matcher.group(1);
        if(averageValue == null) {
            logger.warn("averageValue is null");
            return "0";
        }
```

Вимірювання пропускної здатності

Iperf3 — це потужний інструмент для тестування пропускної здатності мережі, який дозволяє виміряти швидкість передачі даних між двома

комп'ютерами. Для проведення тесту потрібно запустити iperf3 в режимі сервера на одному комп'ютері, а на іншому — в режимі клієнта. Сервер очікує з'єднання на стандартному порту 5201 (або іншому, за вказівкою), а клієнт ініціює з'єднання, після чого розпочинається вимірювання пропускної здатності. Інструмент підтримує як TCP, так і UDP протоколи, що дозволяє проводити різні види тестування, включаючи оцінку затримок і втрат пакетів для UDP-з'єднань.

Використовуючи iperf3, можна налаштовувати різні параметри тесту, такі як тривалість вимірювання, кількість потоків для тестування та пропускну здатність для UDP-з'єднань. Результати тесту відображаються у вигляді статистики, що включає середню швидкість передачі даних, затримки, а для UDP — втрати пакетів. Це дає змогу точно оцінити продуктивність мережі і визначити її слабкі місця. Завдяки своїй простоті та гнучкості, iperf3 є одним із найпопулярніших інструментів для мережевих адміністраторах і інженерів.

Для вимірювання пропускної здатності мережі за допомогою утиліти iperf3 спочатку необхідно встановити цю програму на сервері [27, 28]. Після успішної інсталяції утиліти на серверній машині потрібно запустити її в режимі сервера, що дозволить утримувати порт, на якому будуть отримуватися з'єднання від клієнтських систем. Сервер буде готовий до проведення тесту, як тільки буде активовано відповідний процес iperf3.

Цей код формує команду для запуску тесту пропускної здатності через утиліту iperf3 на задану IP-адресу, використовуючи форматування рядка.

Нижче наведено приклад коду, який запускає тест пропускної здатності через утиліту iperf3 на задану IP-адресу. Потім команда виконується в системі, і зчитуються результати її виконання. За допомогою потоку вводу зчитується кожен рядок виводу програми, який додається до змінної, що зберігає результат. Якщо виникає помилка при виконанні команди, вона обробляється через виключення, і виводиться попередження. В кінці закривається потік вводу.

```
StringBuilder outputResult = new StringBuilder();
String iperf3Cmd = String.format("iperf3 -c %d", ip);
try {
```

```
          Runtime runtime = Runtime.getRuntime();
          Process process = runtime.exec(iperf3Cmd);
          InputStreamReader        streamReader        =        new
InputStreamReader(process.getInputStream());
          BufferedReader in = new BufferedReader(streamReader);
          String inputLine;
          while ((inputLine = in.readLine()) != null) {
               outputResult.append(inputLine);
          }
          in.close();
     } catch (IOException e) {
          logger.warn("createBandwidth exception ", e);
     }
```

Вимірювання втрат пакетів за допомогою утиліти ping з параметром -c 100 дозволяє оцінити якість з’єднання та визначити, чи є проблеми з доставкою даних через мережу. Параметр -c 100 вказує утиліті відправити 100 ICMP запитів до вказаного хоста, що дає змогу отримати статистику про можливі втрати пакетів протягом серії перевірок. Це дозволяє оцінити стабільність з'єднання в умовах більше тривалого тестування, що знижує ймовірність випадкових коливань і дає точніше уявлення про ефективність мережі.

```
     StringBuilder outputResult = new StringBuilder();
     String ping3Cmd = String.format("ping -c 100 %d", ip);
     try {
          Runtime runtime = Runtime.getRuntime();
          Process process = runtime.exec(ping3Cmd);
          InputStreamReader        streamReader        =        new
InputStreamReader(process.getInputStream());
          BufferedReader in = new BufferedReader(streamReader);
          String inputLine;
          while ((inputLine = in.readLine()) != null) {
               outputResult.append(inputLine);
          }
          in.close();
```

Після вимірювання затримки, пропускної здатності та втрат пакетів всі дані були записані у файл data.csv для подальшого аналізу.

3.2 Розробка базової моделі машинного навчання

Для автоматизованого вибору протоколів було розроблено модель на основі машинного навчання [30-33]. Метод ґрунтується на застосуванні рекурентних нейронних мереж (RNN) з використанням LSTM для обробки часових рядів, що дозволяє передбачити мережеві умови та перемикати протоколи в реальному часі (Додаток А.). Модель враховує такі параметри, як затримка, пропускна здатність і втрати пакетів (Додаток Б).

Основні етапи реалізації:

1. Збір даних: Використання утиліт ping та iperf3 для аналізу мережевого трафіку та Ant Media Server для тестування протоколів.
2. Побудова моделі: Використання Python з бібліотеками TensorFlow та scikit-learn для навчання нейронної мережі.
3. Реалізація алгоритму вибору: На основі прогнозу мережевих параметрів модель обирає оптимальний протокол для забезпечення якісного стрімінгу (Додаток В).

Під час передачі аудіо- та відеосигналів від джерела на сервер, де вони обробляються і доставляються кінцевим користувачам через відповідні протоколи, такі як RTMP або WebRTC, забезпечується можливість їх перегляду. RTMP зазвичай використовується для доставки контенту на сервери, забезпечуючи стабільність передачі та сумісність із платформами. WebRTC, у свою чергу, оптимальний для забезпечення прямої трансляції завдяки мінімальній затримці [34-36]. Основні вхідні параметри можуть включати:

- Затримка мережі (ms):, 100 мс.
- Пропускна здатність (Mbps):, 5 Mbps.
- Втрати пакетів (%): 0.5%.

Для визначення оптимального протоколу передачі даних, такого як WebRTC або RTMP, у схожих умовах можна застосувати кластеризацію [37, 38]. Цей підхід дозволяє системі адаптуватися до динамічних змін мережевих параметрів та автоматично обирати протокол, оптимальний для конкретного сценарію.

```
from sklearn.cluster import KMeans
from pandas import read_csv
import numpy as np

path = r"data.csv"
names = ['date_time', 'latency', 'bandwidth', 'packet_loss']

dataframe = read_csv(path, names = names, header=0)

data = dataframe[[ 'latency', 'bandwidth', 'packet_loss'
]].values

# Кластеризація на 2 групи (WebRTC та RTMP)
kmeans = KMeans(n_clusters=2, random_state=0).fit(data)

# Прогноз кластеру для нових даних
new_data = np.array([[100, 5, 0]])
cluster = kmeans.predict(new_data)

protocol = 'RTMP'  if cluster == 1 else 'WebRTC'
print(f"Рекомендований протокол: {protocol}")
```

Програма визначила, що при хороших умовах мережі, таких як висока пропускна здатність і низькі втрати пакетів, доцільно використовувати RTMP, який забезпечує стабільність і високу якість відео. У випадках з гіршою якістю з'єднання (висока затримка або втрати пакетів) програма рекомендує використовувати WebRTC, що адаптується до динамічних умов, хоча й може знижувати якість відео для мінімізації затримки.

Для забезпечення ефективної роботи методу вибору між WebRTC та DASH необхідно збирати точні дані про мережеві умови. У цьому процесі важливу роль відіграють Wireshark і Ant Media Server. Wireshark використовується для аналізу мережевого трафіку, що включає вимірювання затримок, втрат пакетів і пропускної здатності в режимі реального часу. Це дозволяє отримати детальну інформацію про стан мережі.

Ant Media Server забезпечує тестове середовище, у якому можна проводити експерименти зі стрімінговими протоколами WebRTC та DASH. Сервер дозволяє вимірювати якість і стабільність потоків у різних мережевих умовах. Платформа генерує корисні дані для навчання моделей машинного навчання, що автоматично перемикають протоколи залежно від стану мережі.

Таке поєднання інструментів гарантує, що система буде максимально точно адаптуватися до змін мережевих параметрів і обирати оптимальний протокол для стабільного передавання мультимедіа.

Особливості потокових протоколів:

- RTMP використовується для передачі потокового контенту до серверів, забезпечуючи надійність і сумісність із багатьма платформами.
- WebRTC забезпечує низьку затримку та підходить для живих трансляцій.
- DASH адаптує якість відео до пропускної здатності мережі, що покращує стабільність передачі.
- HLS використовується в екосистемі Apple, хоча інші компанії також працюють над його сумісністю.
- MPEG-DASH позиціонується як універсальний стандарт, проте ще не повністю підтримується Apple.

Для реалізації такого підходу збираються дані про параметри мережі в реальному часі. Вхідні параметри можуть бути такими:

- Затримка мережі (ms):, 50 мс.
- Пропускна здатність (Mbps):, 5 Mbps.
- Втрати пакетів (%): 2%.
- Тип контенту: пряма трансляція або відео на вимогу (VOD).

– Кількість підключених користувачів: 100 користувачів.

Реалізація коду, що здійснює прогнозування оптимального протоколу для використання — WebRTC чи DASH — на основі вхідних даних, представлена в Додатку А.

1. Дані: У прикладі мережеві параметри представлені у вигляді масиву, де кожен рядок містить затримку, пропускну здатність і втрати пакетів.

2. Модель: Використовується проста нейронна мережа з двома прихованими шарами, щоб передбачити, який протокол краще обрати.

3. Результат: На основі нових параметрів мережі модель прогнозує, який протокол обрати: WebRTC або DASH.

Цей підхід дозволяє автоматизувати процес вибору протоколу на основі реальних мережевих умов, забезпечуючи оптимальний досвід для користувачів.

3.3 Аналіз та оцінка результатів реалізації

Отримавши реальні дані про мережевий трафік можна буде розглянути і провести порівняння різних варіантів аналізу та оптимізації роботи стрімінгових протоколів, зокрема:

1. Вибір протоколу для передачі аудіо- та відео на сервер за умов затримки 100 мс, пропускної здатності 5 Mbps і втрат пакетів 0.5%: оптимальним є RTMP для стабільної трансляції, проте WebRTC підходить для мінімізації затримки, хоча може знижувати якість відео.

2. Вибір протоколу на основі порогових значень. Якщо затримка менше 100 мс та втрати пакетів нижчі за 2%, обирається DASH інакше –WebRTC.

3. Регресійний аналіз. Використання лінійної чи поліноміальної регресії для прогнозування пропускної здатності та затримок у майбутньому.

4. Кластеризація. Використання алгоритмів, таких як K-Means, для групування мережевих умов та автоматичного налаштування протоколів для кожного кластера.

5. Рекомендаційна система. Створення моделі, що враховує історичні дані користувача (часові зони, попередні затримки) для персоналізованого вибору протоколу.

Вибір оптимального підходу залежить від поставлених цілей. Якщо мета – швидкий аналіз даних у реальному часі, краще використовувати порогові значення або регресію, оскільки вони швидко обробляють дані й легко інтегруються.

Однак для більш складних сценаріїв, де мережеві умови змінюються динамічно, кластеризація може виявитися корисною. Вона дозволяє виявляти закономірності та автоматично адаптувати систему до нових умов без необхідності жорстко запрограмованих правил. Обидва підходи можна комбінувати для більшої ефективності.

Щоб визначити оптимальні протоколи для схожих умов без складних правил можна використати кластеризацію. Цей підхід дає можливість адаптувати систему до динамічних умов та оптимізувати вибір протоколу для різних сценаріїв.

```
from sklearn.cluster import KMeans
from pandas import read_csv
import numpy as np

# Приклад зібраних даних: затримка, пропускна здатність,
втрати пакетів
path = r"data.csv"
names = ['date_time', 'latency', 'bandwidth', 'packet_loss']

dataframe = read_csv(path, names = names)

data = dataframe[[ 'latency', 'bandwidth', 'packet_loss'
]].values

# Кластеризація на 2 групи (умовно: RTMP та DASH)
kmeans = KMeans(n_clusters=2, random_state=0).fit(data)

# Прогноз кластеру для нових даних
new_data = np.array([[80, 5, 2]])
cluster = kmeans.predict(new_data)

protocol = 'DASH'  if cluster == 0 else 'WebRTC'
print(f"Рекомендований протокол: {protocol}")
```

Висновки до розділу 3

1. Показано як реалізація потокової передачі забезпечує адаптивний вибір протоколу для аудіо- та відеотрансляцій. У процесі використання архітектури застосовуються RTMP та WebRTC, які дають змогу досягти балансу між якістю трансляції та низькою затримкою. WebRTC демонструє себе як оптимальний протокол для інтерактивних трансляцій завдяки мінімальній затримці, тоді як DASH забезпечує стабільність відтворення навіть за умов нестабільного з'єднання.

2. Для вимірювання параметрів мережі були використані утиліти ping та iperf3. З їхньою допомогою вдалося ефективно визначити затримку, втрати пакетів і пропускну здатність мережі. Утиліта ping показала високу ефективність у швидкій оцінці затримок, а iperf3 надала можливість детально аналізувати швидкість передачі даних між сервером і клієнтом.

3. У процесі роботи була розроблена модель машинного навчання для автоматизованого вибору між протоколами WebRTC і DASH. Ця модель базується на рекурентних нейронних мережах (LSTM), які обробляють часові ряди мережевих параметрів. Використання реальних даних, отриманих через Wireshark та Ant Media Server, дозволило налаштувати модель на роботу в реальних мережевих умовах.

4. Під час аналізу особливостей протоколів потокової передачі було встановлено, що WebRTC підходить для сценаріїв із низькою затримкою, таких як відеоконференції або інтерактивні трансляції. У той же час DASH виявився кращим для забезпечення стабільного відтворення відео завдяки адаптації якості до пропускної здатності мережі.

5. Для аналізу та оптимізації алгоритмів вибору використовувалися кілька підходів: порогові значення, регресійний аналіз, кластеризація та рекомендаційні системи. З'ясувалося, що для динамічних умов мережі кластеризація є найефективнішою, оскільки вона автоматично адаптує вибір протоколів, враховуючи змінювані умови.

6. Зібрані дані про мережеві параметри були записані у файл data.csv, що дозволяє зберігати історію для подальшого аналізу. Це створює базу для покращення моделі в майбутньому, зокрема для її адаптації до нових умов.

7. У результаті досліджень визначено рекомендації для подальшого вдосконалення. Зокрема, пропонується інтегрувати кластеризацію з нейронною мережею для підвищення точності прогнозів і розширити тестування моделі на ширший спектр мережевих умов, що зробить її більш універсальною.

ВИСНОВКИ

1. Ant Media Server забезпечує високу ефективність у передаванні відео та аудіо з низькою затримкою, що робить його ідеальним вибором для реального часу трансляцій. Завдяки підтримці таких протоколів, як RTMP та DASH, ця платформа здатна забезпечити стабільний і якісний потік навіть у складних мережевих умовах. Це робить її зручним інструментом для тестування різних сценаріїв та оптимізації потокової передачі з урахуванням змінних параметрів мережі.

2. Сучасні технології машинного навчання відкривають нові можливості, дозволяючи значно підвищити точність прогнозів завдяки здатності працювати з великими обсягами даних. За допомогою таких методів, як глибинне навчання та нейронні мережі, можливо виявляти складні патерни та взаємозв'язки в історичних даних ринку, що забезпечує точніші прогнози та більш ефективне прийняття рішень. Алгоритми машинного навчання здатні автоматично адаптуватися до змінюваних умов ринку, дозволяючи фінансовим аналітикам та інвесторам знижувати ризики і покращувати стратегії управління активами.

3. Модель LSTM є одним з найбільш підходящих інструментів для прогнозування та аналізу часових рядів у контексті змінних мережевих умов. Вона дозволяє ефективно передбачати варіації таких параметрів, як пропускна здатність та затримка, що є критичними для вибору оптимального протоколу в реальному часі. LSTM допомагає зберігати важливу інформацію з попередніх проміжків часу, що робить її особливо корисною для прогнозування та оптимізації процесів потокової передачі.

4. Інтеграція технологій машинного навчання та автоматизації створює потужні можливості для розробки гнучких рішень, які здатні ефективно враховувати складність та мінливість ринкових умов. Завдяки здатності машинного навчання до самонавчання та адаптації, ці системи можуть оперативно реагувати на зміни в економічному середовищі, що дозволяє не лише точніше прогнозувати майбутні тренди, але й автоматично коригувати стратегії в реальному часі. Такий підхід значно підвищує ефективність управління

фінансовими активами, оптимізуючи процеси прийняття рішень та знижуючи ризики, що виникають через непередбачуваність ринку.

5. Проведений аналіз сучасних методів і технологій потокової передачі показав, що ключовими факторами для забезпечення якісного стрімінгу є коротка затримка запуску, плавне відтворення та адаптація до мережевих умов. Найбільш перспективними є адаптивні технології, такі як MPEG-DASH і HLS, які динамічно змінюють якість потоку в залежності від доступної пропускної здатності мережі.

6. Традиційні протоколи, такі як RTSP і RTMP, хоча і забезпечують низьку затримку та стабільність передачі, мають обмеження у масштабованості та підтримці сучасних пристроїв. Крім того, їхня залежність від спеціалізованих серверів та проблеми з проходженням через NAT і брандмауери ускладнюють їх використання в глобальних масштабах.

7. Прогресивне завантаження HTTP показало свою простоту реалізації та сумісність із різними пристроями. Однак обмежена функціональність, зокрема неможливість адаптувати якість відео до змін у мережі, робить цей підхід менш привабливим для сучасних динамічних платформ.

8. Адаптивна потокова передача через HTTP (наприклад, HLS, MPEG-DASH) виділяється своєю масштабованістю та гнучкістю. Використання CDN, підтримка шифрування та функції динамічного регулювання якості забезпечують оптимальний користувацький досвід навіть за складних мережевих умов.

9. Аналіз реалізацій таких протоколів, як RTMP, Smooth Streaming, HLS і MPEG-DASH, підтвердив їхню актуальність у різних сценаріях. Зокрема, RTMP залишається популярним для прийому трансляцій, тоді як HLS і MPEG-DASH демонструють високу ефективність у доставці контенту кінцевим користувачам.

10. Використання CDN є важливим елементом у сучасних потокових платформах. Завдяки розподілу навантаження та мінімізації затримок, CDN забезпечує швидку та стабільну доставку контенту навіть при високих навантаженнях, таких як глобальні трансляції або популярні події.

11. MPEG-DASH підтвердив свою універсальність та ефективність завдяки інтеграції з існуючими мережами та підтримці функцій адаптивної передачі. Можливість зміни якості потоку, інтеграція DRM і підтримка різних профілів роблять його провідним стандартом у сфері потокової передачі.

12. Загальний висновок аналізу полягає в тому, що вибір технології повинен враховувати специфічні вимоги платформи, такі як підтримка пристроїв, масштабованість і адаптація до мережевих умов. Адаптивні протоколи, такі як MPEG-DASH, представляють оптимальне рішення для забезпечення якісного і гнучкого стрімінгу в сучасних умовах.

13. Показано як реалізація потокової передачі забезпечує адаптивний вибір протоколу для аудіо- та відеотрансляцій. У процесі використання архітектури застосовуються RTMP та WebRTC, які дають змогу досягти балансу між якістю трансляції та низькою затримкою. WebRTC демонструє себе як оптимальний протокол для інтерактивних трансляцій завдяки мінімальній затримці, тоді як DASH забезпечує стабільність відтворення навіть за умов нестабільного з’єднання.

14. Для вимірювання параметрів мережі були використані утиліти ping та iperf3. З їхньою допомогою вдалося ефективно визначити затримку, втрати пакетів і пропускну здатність мережі. Утиліта ping показала високу ефективність у швидкій оцінці затримок, а iperf3 надала можливість детально аналізувати швидкість передачі даних між сервером і клієнтом.

15. У процесі роботи була розроблена модель машинного навчання для автоматизованого вибору між протоколами WebRTC і DASH. Ця модель базується на рекурентних нейронних мережах (LSTM), які обробляють часові ряди мережевих параметрів. Використання реальних даних, отриманих через Wireshark та Ant Media Server, дозволило налаштувати модель на роботу в реальних мережевих умовах.

16. Під час аналізу особливостей протоколів потокової передачі було встановлено, що WebRTC підходить для сценаріїв із низькою затримкою, таких як відеоконференції або інтерактивні трансляції. У той же час DASH виявився

кращим для забезпечення стабільного відтворення відео завдяки адаптації якості до пропускної здатності мережі.

17. Для аналізу та оптимізації алгоритмів вибору використовувалися кілька підходів: порогові значення, регресійний аналіз, кластеризація та рекомендаційні системи. З’ясувалося, що для динамічних умов мережі кластеризація є найефективнішою, оскільки вона автоматично адаптує вибір протоколів, враховуючи змінювані умови.

18. Зібрані дані про мережеві параметри були записані у файл data.csv, що дозволяє зберігати історію для подальшого аналізу. Це створює базу для покращення моделі в майбутньому, зокрема для її адаптації до нових умов.

19. У результаті досліджень визначено рекомендації для подальшого вдосконалення. Зокрема, пропонується інтегрувати кластеризацію з нейронною мережею для підвищення точності прогнозів і розширити тестування моделі на ширший спектр мережевих умов, що зробить її більш універсальною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. YouTube. URL: https://www.youtube.com
2. Netflix. URL: https://www.netflix.com
3. Twitch. URL: https://www.twitch.tv
4. Li, X., & Mao, Y. (2021). Adaptive streaming with DASH: A survey on challenges and solutions. IEEE Communications Surveys & Tutorials.
5. Adobe Systems Incorporated. (2020). Real-Time Messaging Protocol (RTMP) Specification.
6. Ant Media Server. URL: https://antmedia.io
7. Kher, R., & Patel, M. (2019). Data imputation techniques in machine learning: A survey. Journal of Big Data.
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
9. Beben A., Batalla J. M., Chai W., Sliwinski J. Multi-criteria decision algorithms for efficient content delivery in content networks. Annals of Telecommunications, 2013, vol. 68, no. 3, pp. 153-165. Springer.
10. Yang, H., Chen, X., Yang, Z., Zhu, X., Chen, Y. (2014). Opportunities and Challenges of HTTP Adaptive Streaming. International Journal of Future Generation Communication and Networking, 7(6), 165–190.
11. Ronny. Creating End-to-End Smooth Streaming Video Solutions with Silverlight and IIS Server.
12. Dash Streaming. MPEG-DASH Overview. Encoding.com. URL: www.encoding.com/mpeg-dash.
13. Weil, N., & Law, W. (2014). MPEG-DASH: Tomorrow's Format with Akamai's Nicolas Weil and Will Law. Edge Presentation.
14. Hossein Sarshar. IIS Smooth Streaming.
15. Apple Developer. (n.d.). HTTP Live Streaming Guide. URL: https://developer.apple.com/library.
16. Vidbeo Blog. (n.d.). HLS vs DASH. URL: www.vidbeo.com/blog/hls-vs-dash.

17. Modi, P. Adaptive Media Streaming: HLS vs MSS vs DASH. URL: digiflare.com/adaptive-media-streaming-hls-vs-mss-vs-dash.

18. Svystun, L. How to Tackle Frequent Live Video Streaming Challenges. Letzgro Blog. URL: https://letzgro.net/blog.

19. Forouzan, B. A. (2017). TCP/IP Protocol Suite (4th ed.). New York: McGraw-Hill.

20. Mueller, C. MPEG-DASH vs. Apple HLS vs. Microsoft Smooth Streaming vs. Adobe HDS. Bitmovin Blog. URL: https://bitmovin.com/mpeg-dash-vs-apple-hls-vs-microsoft-smooth-streaming-vs-adobe-hds.

21. Ozer, J.. What is HTTP Live Streaming? Streaming Media. URL: www.streamingmedia.com

22. Narang, N.. (2015). Concept Series: Progressive Download, RTMP Streaming and Adaptive Streaming. URL: www.mediaentertainmentinfo.com.

23. Adhikari V. K., Jain S., Chen Y., Zhang Z. Vivisecting YouTube: An Active Measurement Study. Technical Report TR 11-012, Department of Computer Science and Engineering, University of Minnesota, 2011.

24. Adhikari V. K., Guo Y., Hao F., Varvello M., Hilt V., Steiner M., Zhang Z.-L. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. IEEE INFOCOM 2012, pp. 1620-1628. IEEE.

25. Gao L. On inferring autonomous system relationships in the Internet. IEEE/ACM Trans. on Networking, 2001, vol. 9, no. 6, pp. 733-745. IEEE.

26. He J., Rexford J. Toward Internet-wide multipath routing. IEEE Network, 2008, vol. 22, no. 2, pp. 16-21. IEEE.

27. Qiu C., Shen H., Chen L. Probabilistic demand allocation for cloud service brokerage // Proc. of INFOCOM. 2016.

28. Yu Y., Jindal F., Bastani V., Li F., Yen I. Improving the smartness of cloud management via machine learning based workload prediction // Proc. of COMPSAC. 2018.

29. Reiss C., Wilkes J., Hellerstein J. Google cluster-usage traces: format+ schema // Google Inc., White Paper. 2011.

30. Calheiros R., Masoumi E., Ranjan R., Buyya R. Workload prediction using arima model and its impact on cloud applications' qos // Trans. on CC. 2015.

31. Shyam G., Manvi S. Virtual resource prediction in cloud environment: a bayesian approach // Journal of Network and Computer Applications. 2016.

32. Chen L., Shen H., Sapra K. Rial: Resource intensity aware load balancing in clouds // Proc. of INFOCOM. 2014.

33. Song B., Yu Y., Zhou Y., Wang Z., Du S. Host load prediction with long short-term memory in cloud computing // Journal of Supercomputing. 2018.

34. Gao, Jiechao, Wang, Haoyu, Shen, Haiying. Machine Learning Based Workload Prediction in Cloud Computing [Text] / J. Gao, H. Wang, H. Shen // Proceedings of the International Conference on Computer Communications and Networks (ICCCN). – 2020. – P. 1–9. – DOI: 10.1109/ICCCN49398.2020.9209730.

35. Wehrle K., Gross J., Gnes M. Modeling and Tools for Network Simulation. Springer, 2010.

36. Готяш Ю.П. Аналіз стрімінгових протоколів для оцінки продуктивності системи потокового відео // Міжнародна мультидисциплінарна наукова інтернет-конференція "Світ наукових досліджень." [Електронний ресурс] – https://www.economy-confer.com.ua/full-article/5891/

37. Готяш Ю.П., Гадевич В.Ю. Аналіз мережевих параметрів для оптимізації вибору стрімінгового протоколу на базі Ant Media Server // Міжнародна науково-практична інтернет-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» [Електронний ресурс] – Режим доступу: http://www.konferenciaonline.org.ua/ua/article/id-2018/

38. Комар М.П., Саченко А.О., Васильків Н.М., Загородня Д.І. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за другим (магістерським) рівнем вищої освіти. – Тернопіль: ЗУНУ, 2024. – 32 с.

Додаток А

Програмна реалізація вибору оптимального протоколу

```
import pandas as pd
import io
import numpy as np
from pandas import read_csv
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM

path = r"data.csv"
names = ['date_time', 'latency', 'bandwidth', 'packet_loss']

dataframe = read_csv(path, names = names)

x_train = dataframe[[ 'latency', 'bandwidth', 'packet_loss'
]].values

# Приклад вхідних даних: [затримка, пропускна здатність, втрати
пакетів]
# x_train = np.array([
#     [50, 5, 2],  # Приклад нормальної мережі
#     [100, 3, 5],  # Приклад поганої мережі
#     [20, 10, 1],  # Відмінна мережа
#     [150, 2, 8]   # Слабке підключення
#])

dataframe = read_csv(path, names = names)

x_train = dataframe[[ 'latency', 'bandwidth', 'packet_loss'
]].values


# Мітки: 0 - DASH, 1 - WebRTC
y = []
for sample in x_train:
latency, bandwidth, packet_loss = sample
if latency > 100 and packet_loss > 5:  # погана мережа
     y.append(1)  # WebRTC
elif bandwidth > 8 and latency < 50:  # хороша мережа
     y.append(0)  # DASH
else:
     y.append(np.random.choice([0, 1]))  # Для інших випадків
випадкова мітка
y_train = np.array(y)

# Створення моделі з LSTM
model = Sequential()
```

```
    model.add(Dense(64,           input_shape=(x_train.shape[1],),
activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # Компіляція моделі
    model.compile(optimizer='adam',    loss='binary_crossentropy',
metrics=['accuracy'])

    # Навчання моделі
    model.fit(x_train, y_train, epochs=10, verbose=1)

    # Приклад тестового прогнозу
    new_data = np.array([[80, 4, 3]])  # Нові мережеві параметри
    prediction = model.predict(new_data)

    # Інтерпретація прогнозу
    protocol = 'DASH'  if prediction < 0.5 else 'WebRTC'
    print(f"Рекомендований протокол: {protocol}")
```

Додаток Б

Результати вимірювання параметрів комп’ютерної мережі

```
2024-11-08T20:33:51.906822157Z,23,44,0
2024-11-08T20:38:51.894746753Z,23,45,0
2024-11-08T20:43:51.894252101Z,23,46,0
2024-11-08T20:48:51.894303383Z,25,49,0
2024-11-08T20:53:51.894288264Z,23,46,0
2024-11-08T20:58:51.894263936Z,23,35,0
2024-11-08T21:03:51.894295841Z,23,47,0
2024-11-08T21:08:51.894262150Z,24,45,0
2024-11-08T21:13:51.894308363Z,22,44,0
2024-11-08T21:18:51.894214185Z,22,46,0
2024-11-08T21:23:51.894221334Z,22,44,0
2024-11-08T21:28:51.894175051Z,22,47,0
2024-11-08T21:33:51.894221804Z,23,48,0
2024-11-08T21:38:51.894279223Z,22,50,0
2024-11-08T21:43:51.894299107Z,23,50,0
2024-11-08T21:48:51.894709683Z,23,48,0
2024-11-08T21:53:51.894186583Z,23,48,0
2024-11-08T21:58:51.894220192Z,23,49,0
2024-11-08T22:03:51.894221963Z,22,49,0
2024-11-08T22:08:51.894277052Z,22,46,0
2024-11-08T22:13:51.894201288Z,22,45,0
2024-11-08T22:18:51.894233763Z,23,49,0
2024-11-08T22:23:51.894240519Z,23,47,0
2024-11-08T22:28:51.894231377Z,23,47,0
2024-11-08T22:33:51.894315308Z,23,48,0
2024-11-08T22:38:51.894226430Z,23,47,0
2024-11-08T22:43:51.894247937Z,23,48,0
2024-11-08T22:48:51.894244619Z,23,49,0
2024-11-08T22:53:51.894231527Z,23,47,0
2024-11-08T22:58:51.894140445Z,26,44,0
2024-11-08T23:03:51.894235445Z,23,49,0
2024-11-08T23:08:51.894265463Z,23,49,0
2024-11-08T23:13:51.894230390Z,22,47,0
2024-11-08T23:18:51.894258482Z,23,47,0
2024-11-08T23:23:51.894313963Z,23,46,0
2024-11-08T23:28:51.894327025Z,23,47,0
2024-11-08T23:33:51.894248927Z,23,47,0
2024-11-08T23:38:51.894261988Z,23,48,0
2024-11-08T23:43:51.894268595Z,23,48,0
2024-11-08T23:48:51.894301639Z,23,48,0
2024-11-08T23:53:51.894171105Z,23,49,0
2024-11-08T23:58:51.894225055Z,22,49,0
2024-11-09T00:03:51.894233188Z,23,48,0
2024-11-09T00:08:51.894314528Z,23,43,0
2024-11-09T00:13:51.894247967Z,23,46,0
2024-11-09T00:18:51.894159003Z,23,46,0
2024-11-09T00:23:51.894228261Z,23,48,0
2024-11-09T00:28:51.894189116Z,23,48,0
2024-11-09T00:33:51.894304957Z,23,46,0
2024-11-09T00:38:51.894223564Z,23,47,0
```

```
2024-11-09T00:43:51.894254059Z,23,48,0
2024-11-09T00:48:51.894159912Z,23,48,1
2024-11-09T00:53:51.894219327Z,23,47,0
2024-11-09T00:58:51.894245698Z,23,45,0
2024-11-09T01:03:51.894223605Z,22,46,0
2024-11-09T01:08:51.894216086Z,23,50,0
2024-11-09T01:13:51.894225432Z,22,49,0
2024-11-09T01:18:51.894256080Z,23,47,0
2024-11-09T01:23:51.894216441Z,23,47,0
2024-11-09T01:28:51.894232813Z,23,47,1
2024-11-09T01:33:51.894254681Z,23,47,0
2024-11-09T01:38:51.894227835Z,23,48,0
2024-11-09T01:43:51.894223106Z,23,50,0
2024-11-09T01:48:51.894271358Z,22,48,0
2024-11-09T01:53:51.894241563Z,23,48,0
2024-11-09T01:58:51.894184412Z,23,46,0
2024-11-09T02:03:51.894182494Z,22,48,0
2024-11-09T02:08:51.894098982Z,23,48,0
2024-11-09T02:13:51.894208541Z,22,50,0
2024-11-09T02:18:51.894186508Z,23,47,0
2024-11-09T02:23:51.894221687Z,23,49,0
2024-11-09T02:28:51.894201506Z,23,49,0
2024-11-09T02:33:51.894192364Z,23,49,0
2024-11-09T02:38:51.894181552Z,22,50,0
2024-11-09T02:43:51.894203163Z,23,48,0
2024-11-09T02:48:51.894202806Z,22,49,0
2024-11-09T02:53:51.894208416Z,22,48,0
2024-11-09T02:58:51.894114439Z,22,49,0
2024-11-09T03:03:51.894181336Z,22,48,0
2024-11-09T03:08:51.894201673Z,23,47,0
2024-11-09T03:13:51.894192604Z,23,48,0
2024-11-09T03:18:51.894168126Z,23,48,0
2024-11-09T03:23:51.894191914Z,23,45,0
2024-11-09T03:28:51.894213498Z,23,48,0
2024-11-09T03:33:51.894221870Z,23,47,0
2024-11-09T03:38:51.894186734Z,23,47,0
2024-11-09T03:43:51.894189192Z,23,48,0
2024-11-09T03:48:51.894185441Z,23,49,0
2024-11-09T03:53:51.894209896Z,22,49,0
2024-11-09T03:58:51.894192731Z,23,48,0
2024-11-09T04:03:51.894118441Z,23,49,0
2024-11-09T04:08:51.894194591Z,23,48,0
2024-11-09T04:13:51.894180391Z,23,49,0
2024-11-09T04:18:51.894188891Z,23,49,0
2024-11-09T04:23:51.894270433Z,21,48,0
2024-11-09T04:28:51.894110008Z,21,48,0
2024-11-09T04:33:51.894209173Z,21,48,0
2024-11-09T04:38:51.894190431Z,21,47,0
2024-11-09T04:43:51.894273113Z,21,47,0
2024-11-09T04:48:51.894173919Z,21,48,0
2024-11-09T04:53:51.894210291Z,21,48,0
2024-11-09T04:58:51.894194161Z,22,46,0
2024-11-09T05:03:51.894178179Z,22,48,0
```

Додаток В

Результати виконання прогнозування

```
D2$ python3 D2Script.py
Epoch 1/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy:
0.2746 - loss: 4.8331
Epoch 2/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8128 - loss: 0.5265
Epoch 3/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy:
0.8166 - loss: 0.3026
Epoch 4/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8423 - loss: 0.2441
Epoch 5/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8263 - loss: 0.2829
Epoch 6/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8001 - loss: 0.2836
Epoch 7/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.7832 - loss: 0.2903
Epoch 8/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8018 - loss: 0.2726
Epoch 9/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8038 - loss: 0.27609
Epoch 10/10
24/24 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8085 - loss: 0.2734
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Рекомендований протокол: DASH
```

Додаток Г

Копії публікацій

АНАЛІЗ СТРІМІНГОВИХ ПРОТОКОЛІВ ДЛЯ ОЦІНКИ ПРОДУКТИВНОСТІ СИСТЕМИ ПОТОКОВОГО ВІДЕО

22.11.2024 15:13
Автор: **Готяш Юрій Павлович, магістрант, Західноукраїнський національний університет**
[2. Інформаційні системи і технології;]

Анотація.

У статті проведено порівняльний аналіз стрімінгових протоколів RTMP, DASH та WebRTC для оцінки продуктивності систем потокового відео в умовах реального часу. Запропоновано використання Ant Media Server як універсального інструменту для тестування та оптимізації стрімінгових рішень.

Постановка проблеми.

Розвиток мультимедійних технологій і популярність онлайн-стрімінгу вимагає використання ефективних протоколів. На даний час популярними є протоколи RTMP (Real-Time Messaging Protocol), DASH (Dynamic Adaptive Streaming over HTTP) та WebRTC (Web Real-Time Communication) які забезпечують якісне передавання даних [1].

Основна проблема полягає у тому, що під час потокової передачі мережеві умови можуть змінюватися в реальному часі (наприклад, зниження пропускної здатності або збільшення затримки) і орієнтація лише на один протокол може призвести до погіршення якості трансляції або виникнення затримок. Тому, необхідний підхід для вибору відповідного протоколу в залежності від параметрів мережі та типу контенту.

Мета і завдання дослідження.

Метою даної роботи є аналіз стрімінгових протоколів для відео та аудіотрансляцій що враховує мережеві умови, типи пристроїв користувачів та вимоги до якості трансляцій, що дає змогу оцінити продуктивність системи потокового відео.

В якості інструменту для тестування потокової передачі в реальних умовах запропоновано використання Ant Media Server [3]. Він підтримує роботу з протоколами RTMP, DASH та WebRTC, тому є ідеальною платформою для їх порівняння для подальшого вибору.

Для досягнення поставленої мети необхідно вирішити такі завдання:

• аналіз сучасних протоколів потокової передачі медіа (RTMP, DASH, WebRTC) та їх особливостей у різних сценаріях;

• розробка алгоритму збору параметрів мережі (пропускна здатність, затримки, втрати пакетів)

• розробка алгоритму аналізу даних та рекомендації вибору оптимального протоколу;

• інтеграція алгоритму з Ant Media Server та налаштування серверного середовища для проведення експериментів;

• тестування та оцінка роботи алгоритму в різних умовах мережі для підтвердження його ефективності;

• підготовка рекомендацій щодо використання розробленого рішення для покращення якості потокової передачі в корпоративних та споживчих додатках;

Виклад основного матеріалу.

Першим завданням для оцінки продуктивності системи потокового відео є аналіз стрімінгових протоколів. В таблиці 1 представлено основні характеристики протоколів RTMP, DASH та WebRTC.

Таблиця 1. Характеристики стрімінгових протоколів.

Характеристика	RTMP	DASH	WebRTC
Тип протоколу	Протокол потокової передачі	Адаптивний протокол потокової передачі	Протокол реального часу (peer-to-peer)
Основне використання	Передача відео на сервери або платформи (наприклад, YouTube, Twitch).	Відео на вимогу (VoD), стрімінг контенту через HTTP.	Відеоконференції, потокове передавання реального часу.
Технологічна база	TCP (з використанням Flash/Adobe)	HTTP/HTTPS	UDP/TCP
Затримка	Відносно низька (2-5 секунд).	Висока (5-15 секунд).	Дуже низька (менше 500 мс).
Підтримка адаптації	Немає автоматичної адаптації до пропускної здатності.	Адаптація до пропускної здатності та роздільної здатності.	Залежить від WebRTC API та реалізації.
Сумісність	Flash (застарілий, але підтримується в OBS та інших інструментах).	Сумісний із більшістю сучасних браузерів та пристроїв.	Нативна підтримка у сучасних браузерах.
Масштабованість	Висока (для великих стрімінгових платформ).	Дуже висока (можливість CDN інтеграції).	Обмежена для великих обсягів одночасних користувачів.
Реалізація	Простий для налаштування.	Складніший через необхідність адаптивного контенту.	Складність налаштування peer-to-peer мереж.
Підтримка інтерактивності	Низька (затримка не дозволяє).	Дуже низька (підходить для трансляцій, але не інтерактиву).	Висока (взаємодія в реальному часі).
Сховище	Погано підходить для зберігання контенту.	Орієнтований на зберігання контенту у сегментах.	Не використовується для зберігання.
Якість відео	Середня/висока залежно від налаштувань.	Дуже висока, завдяки адаптації.	Залежить від мережі, зазвичай середня/висока.
Підтримка мобільних пристроїв	Підтримка через сторонні додатки та SDK.	Нативна підтримка через браузери.	Нативна підтримка через браузери.
Безпека	Підтримує SSL/TLS (RTMPS).	HTTPS забезпечує безпеку.	Вбудована шифрація (DTLS, SRTP).

Провівши аналіз характеристик протоколів стрімінгу можна зробити наступні висновки.

RTMP залишається популярним для передачі відео від клієнта до сервера, особливо для стрімінгових платформ. Однак його затримка (2-5 секунд) і відсутність адаптації до пропускної здатності обмежують його використання в сучасних інтерактивних та масштабованих системах. Він поступово витісняється іншими протоколами через застарілість Flash.

DASH є сучасним адаптивним протоколом, орієнтованим на потокове передавання через HTTP. Він забезпечує високу якість відео, масштабованість та підтримку адаптації до різної пропускної здатності. DASH найкраще підходить для відео на вимогу (VoD) та великих стрімінгових платформ із CDN. Однак його висока затримка (5-15 секунд) робить його непридатним для застосувань у реальному часі.

WebRTC є найбільш підходящим протоколом для додатків реального часу, таких як відеоконференції, інтерактивні стріми чи ігрові сервіси. Його головна перевага — дуже низька затримка (менше 500 мс) та висока інтерактивність. Водночас WebRTC має обмеження у масштабованості для великої кількості користувачів і потребує складного налаштування peer-to-peer з'єднань.

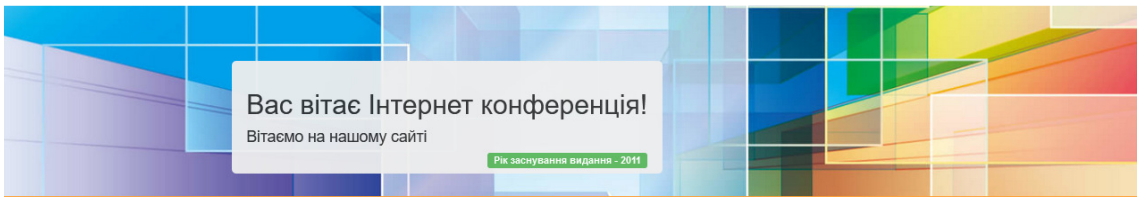
Висновки.

Враховуючи зібрані дані можна зробити наступні рекомендації по вибору відповідного протоколу. Для забезпечення низької затримки та інтерактивності використовується протокол WebRTC. Для масштабованих трансляцій або відео на вимогу, використовується протокол DASH з інтеграцією CDN. Для простих потокових трансляцій використовується RTMP, особливо при використанні OBS або інших стрімінгових інструментів. Використання Ant Media Server дає можливість інтегрувати всі три протоколи, дозволяючи створювати універсальні стрімінгові рішення для різних сценаріїв.

Список літератури

1. Yang H., Chen X., Yang Z., Zhu X., Chen Y. Opportunities and Challenges of HTTP Adaptive Streaming // International Journal of Future Generation Communication and Networking. 2014. №7(6). С. 165–190.

2. Ant Media. Ant Media Server Documentation [Електронний ресурс]. URL: https://antmedia.io (дата звернення: 15.11.2024).



ІМПУТАЦІЯ ПРОПУЩЕНИХ ДАНИХ ЗА ДОПОМОГОЮ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ ТА НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

[1. Інформаційні системи і технології]

Автор: **Гончар Ярослав Андрійович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

АНАЛІЗ МЕРЕЖЕВИХ ПАРАМЕТРІВ ДЛЯ ОПТИМІЗАЦІЇ ВИБОРУ СТРІМІНГОВОГО ПРОТОКОЛУ НА БАЗІ ANT MEDIA SERVER

[1. Інформаційні системи і технології]

Автор: **Готяш Юрій Павлович, магістрант, Західноукраїнський національний університет, м. Тернопіль; Гадевич Володимир Юрійович, магістрант, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

АВТОМАТИЗОВАНА СИСТЕМА ПРОГНОЗУВАННЯ ЕФЕКТИВНОСТІ ПРИВАТНОЇ СОНЯЧНОЇ ЕЛЕКТРОСТАНЦІЇ

[1. Інформаційні системи і технології]

Автор: **Григоренко Олег Ігорович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

РОЗПОДІЛЕНЕ ГЛИБОКЕ НАВЧАННЯ ДЛЯ ОБРОБКИ ДАНИХ ІНТЕРНЕТУ РЕЧЕЙ

[1. Інформаційні системи і технології]

Автор: **Демчишин Владислав Володимирович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

ІНТЕГРАЦІЯ GOOGLE SHEETS З PYTHON ТА MATLAB ДЛЯ СКЛАДНИХ ОБЧИСЛЕНЬ

[1. Інформаційні системи і технології]

Автор: **Замуруєва О.В., кандидат фізико-математичних наук, доцент кафедри теоретичної та комп'ютерної фізики імені А. В. Свідзинського Навчально-наукового фізико-технологічного інституту Волинського національного університету імені Лесі Українки, м. Луцьк; Шваліковський А., студент 1-ого курсу Навчально-наукового фізико-технологічного інституту Волинського національного університету імені Лесі Українки, м. Луцьк; Мельничук А., студент 1-ого курсу Навчально-наукового фізико-технологічного інституту Волинського національного університету імені Лесі Українки,**

***Готяш Юрій Павлович**, магістрант,*
Західноукраїнський національний університет, м. Тернопіль
***Гадевич Володимир Юрійович**, магістрант,*
Західноукраїнський національний університет, м. Тернопіль
Науковий керівник: Биковий Павло Євгенович,
кандидат технічних наук, доцент,
Західноукраїнський національний університет, м. Тернопіль

АНАЛІЗ МЕРЕЖЕВИХ ПАРАМЕТРІВ ДЛЯ ОПТИМІЗАЦІЇ ВИБОРУ СТРІМІНГОВОГО ПРОТОКОЛУ НА БАЗІ ANT MEDIA SERVER

Анотація

У статті описано методики вимірювання ключових параметрів комп’ютерної мережі, зокрема затримки, пропускної здатності та втрат пакетів. Детально розглянуто реалізацію цих вимірювань на базі Ant Media Server і проведено аналіз отриманих даних. Отримані результати є вхідними даними для системи автоматичного вибору протоколу, залежно від вимог до якості, затримки та стабільності з'єднання.

Постановка задачі

Пряма (Live) трансляція полягає в процесі передачі аудіо- та відеосигналів від джерела на сервер, де вони обробляються та передаються кінцевим користувачам. Для передачі сигналів від джерела до сервера зазвичай використовується протокол RTMP. Протокол WebRTC забезпечує мінімальну затримку при передачі даних безпосередньо кінцевим користувачам у режимі реального часу. Протокол DASH дозволяє адаптувати якість відео до змінних мережевих умов, що забезпечує стабільність відтворення, але має більшу затримку. У таких умовах автоматичний вибір протоколу, враховуючи

різноманітність мережевих підключень та пристроїв кінцевих користувачів, є актуальною задачею [1].

Вимірювання затримки

Вимірювання затримки здійснюється за допомогою утиліти ping, яка є одним із найефективніших способів оцінити час, необхідний для передачі пакету даних між двома точками в мережі. Утиліта ping надсилає ICMP-запит до вказаного хоста та вимірює час, що проходить від моменту відправлення запиту до отримання відповіді. Це дозволяє визначити затримку у мілісекундах, оцінити якість мережевого з'єднання та виявити можливі проблеми, такі як висока затримка або втрати пакетів. Затримка до 50 мс вважається гарним показником для більшості стрімінгових застосунків, тоді як значення понад 100 мс можуть свідчити про потенційні проблеми з мережевим з'єднанням.

Вимірювання пропускної здатності

Iperf3 — це потужний та гнучкий інструмент для тестування пропускної здатності мережі, який дозволяє вимірювати швидкість передачі даних між двома пристроями. Для проведення тесту необхідно запустити iperf3 у режимі сервера на одному комп’ютері та в режимі клієнта на іншому. Сервер, зазвичай, працює на стандартному порту 5201 (або іншому, якщо вказано), очікуючи підключення клієнта, який ініціює тестування.

Інструмент підтримує як TCP, так і UDP протоколи, що дозволяє виконувати різні типи тестів, включаючи оцінку затримки та втрат пакетів для UDP-з’єднань. Використовуючи iperf3, можна налаштовувати різні параметри тесту, такі як тривалість вимірювання, кількість потоків передачі даних, пропускну здатність для UDP-з’єднань.

Результати тесту відображаються у вигляді статистики, яка включає середню швидкість передачі даних, затримки та, для UDP, втрати пакетів. Ця інформація дозволяє точно оцінити продуктивність мережі та виявити її слабкі місця. Завдяки своїй простоті та широким можливостям налаштування, iperf3 є

одним із найпопулярніших інструментів для мережевих адміністраторів і інженерів.

Для використання iperf3 спочатку потрібно встановити програму на серверній машині. Після інсталяції сервер запускається в режимі очікування з'єднань клієнта. Як тільки сервер активується, він готовий до тестування.

Вимірювання втрат пакетів

Вимірювання втрат пакетів здійснюється за допомогою утиліти ping з параметром -c 100 дозволяє оцінити якість з'єднання та визначити, чи є проблеми з доставкою даних через мережу. Параметр -c 100 вказує утиліті відправити 100 ICMP запитів до вказаного хоста, що дає змогу отримати статистику про можливі втрати пакетів протягом серії перевірок. Це дозволяє оцінити стабільність з'єднання в умовах більше тривалого тестування, що знижує ймовірність випадкових коливань і дає точніше уявлення про ефективність мережі.

Результати

Для вимірювання був використаний Ant Media Server, як платформа для трансляції відео в реальному часі, яка підтримує популярні срімінг протоколи RTMP, HLS, DASH, WebRTC [2]. Цей сервер оптимізовано для забезпечення високоякісного відео стрімінгу з низькою затримкою, що робить його ідеальним для таких додатків, як відеоконференції, ігрові стріми, навчальні платформи та інші інтерактивні трансляції.

Після вимірювання затримки, пропускної здатності та втрат пакетів всі дані були записані у файл data.csv для подальшого аналізу. Частина даного файлу представлена нижче:

```
2024-11-10 T15:43:51.894072597Z,32,1,15
2024-11-10 T15:48:51.894036837Z,41,1,21
2024-11-10 T15:53:51.894124444Z,28,0,23
2024-11-10 T15:58:51.894158470Z,291,1,22
2024-11-10 T16:03:51.894053477Z,166,1,9
2024-11-10 T16:08:51.894161324Z,42,2,4
2024-11-10 T16:13:51.894109856Z,38,1,11
2024-11-10 T16:18:51.894125589Z,25,2,4
2024-11-10 T16:23:51.894146960Z,32,2,17
2024-11-10 T16:28:51.894116286Z,48,1,6
2024-11-10 T16:33:51.894153890Z,24,1,5
2024-11-10 T16:38:51.894178865Z,36,1,7
2024-11-10 T16:43:51.894116818Z,27,1,10
```

В ході дослідження показано, що WebRTC забезпечує мінімальну затримку, що особливо важливо для інтерактивних трансляцій і відеозв'язку. Водночас DASH дозволяє адаптувати якість відео до змінних мережевих умов, забезпечуючи стабільне відтворення навіть за нестабільного з'єднання, проте він є повільніший. На відміну від HLS, який через буферизацію має вищу затримку, ці протоколи значно покращують досвід користувачів, пропонуючи більш гнучкі та ефективні рішення для різних сценаріїв використання.

Висновок

У даній статті проведено аналіз якості мережевих з'єднань для забезпечення прямої трансляції з використанням Ant Media Server, який забезпечує який високу продуктивність і гнучкість у роботі з стрімінг протоколами. Дослідження охопило вимірювання затримки, пропускної здатності та втрат пакетів із використанням утиліт ping та iperf3, що дозволило оцінити стабільність і ефективність мережевої інфраструктури. Результати дослідження дозволи оцінити реальні умови експлуатації, проаналізувати ключові показники мережі для подальшого вибору протоколу для різних сценаріїв використання. Результати тестувань підтвердили доцільність автоматичного вибору протоколу залежно від потреб конкретної трансляції, що дозволяє забезпечити оптимальну якість обслуговування для кінцевих користувачів.

Література:

1. Letzgro. How to tackle frequent live video streaming challenges [Електронний ресурс]. – Режим доступу: https://letzgro.net/how-to-tackle-frequent-live-video-streaming-challenges/
2. Ant Media. Ant Media Server Documentation [Електронний ресурс]. URL: https://antmedia.io.