НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий Інститут телекомунікаційних систем

Кафедра електронних комунікацій та інтернету речей

«До захисту допущено»
ВО завідувача кафедри
_____ В’ячеслав НОСКОВ
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Аналіз протоколів передачі даних у мережах електронних комунікацій для мультимедійного мовлення прямих трансляцій»

Виконав (-ла):
студент (-ка) IV курсу, групи ТС-12
Костюк Володимир Артемович _____

Керівник:
Старший викладач, к.т.н.
Новіков В.І. _____

Рецензент:
Професор кафедри ТК, д.т.н., професор
Лисенко О.І. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий Інститут телекомунікаційних систем
Кафедра електронних комунікацій та інтернету речей

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 Телекомунікації та радіотехніка

Освітня програма – «Системи електронних комунікацій та інтернету речей»

ЗАТВЕРДЖУЮ
ВО завідувача кафедри
_____ В’ячеслав НОСКОВ
«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Костюку Володимиру Артемовичу

1. Тема роботи «Аналіз протоколів передачі даних у мережах електронних комунікацій для мультимедійного мовлення прямих трансляцій», керівник роботи Новіков Валерій Іванович, к.т.н., затверджені наказом по університету від 26 травня 2025 р. № 1755 – с

2. Термін подання студентом роботи 10 червня 2025 року

3. Вихідні дані до роботи: матеріали статей та наукових видань, інформаційні ресурси мережі Інтернет, навчально-методичні матеріали. Структурований план порядку розробки матеріалів дипломної роботи.

4. Зміст роботи: Обгрунтувати актуальність теми. Розглянути сучасні протоколи передавання даних для мультимедійного мовлення. Провести класифікацію та порівняльний аналіз протоколів TCP, UDP, SRT, QUIC, HLS, DASH та WebRTC за ключовими параметрами. Надати рекомендації щодо вибору протоколів залежно від технічних умов трансляції. Проаналізувати вплив технологій 5G, edge computing та штучного інтелекту на ефективність передавання мультимедійного трафіку. Навести практичні приклади інтеграції протоколів у сучасні трансляційні платформи.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):) Тема та мета дипломної роботи; 2)Актуальність теми; 3) Мета та

завдання роботи; 4) Класифікація протоколів передачі даних; 5) Принцип встановлення з’єднання; 6) Порівняння протоколів TCP, UDP, SRT, HLS, WebRTC ; 7) Сценарії вибору протоколів для трансляцій; 8) Інтеграція стрімінгових протоколів у хмарні сервіси; 9) Вплив новітніх технологій на мультимедійне мовлення; 10) Висновки

6. Дата видачі завдання 31.11.2024 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Збір вивчення рекомендованої літератури	2.11.24 - 19.12.24	Виконано
2	Робота з 1 розділом	19.12.24 – .14.01.25	Виконано
3	Робота з 2 розділом	14.01.25 – 08.02.25	Виконано
4	Робота з 3 розділом	08.02.25 – 10.03.25	Виконано
5	Робота з 4 розділом	10.03.25 – 15.04.25	Виконано
6	Оформлення пояснювальної записки, презентації до захисту	15.04.25 – 24.05.25	Виконано
7	Подання роботи на кафедру	24.05.25 - 10.06.25	Виконано

Студент Володимир КОСТЮК

Керівник роботи Валерій НОВІКОВ

РЕФЕРАТ

Текстова частина дипломної роботи: 123 с., 5 рис., 14 табл., 19 джерел.

Мета роботи — аналіз протоколів передачі даних у мережах електронних комунікацій з метою забезпечення ефективної мультимедійної трансляції у режимі реального часу, визначення їх переваг, недоліків та оптимального використання залежно від умов трансляції.

У роботі досліджено основні характеристики протоколів TCP, UDP, SRT, WebRTC, HLS, MPEG-DASH та QUIC, здійснено порівняльний аналіз їх функціональності, затримки, адаптивності, масштабованості, надійності та ресурсозатратності. Проаналізовано критерії оцінювання ефективності, зокрема затримку, джиттер, втрати пакетів, якість відео та аудіо, безпеку та сумісність.Запропоновано рекомендації щодо вибору протоколу відповідно до типу трансляції (VoD, live, інтерактив), технічних обмежень та сучасних вимог, з урахуванням впливу технологій 5G, edge computing та AI-адаптації. Обґрунтовано доцільність використання гібридних архітектур і сучасних протоколів для підвищення якості стрімінгу у кіберспортивних, освітніх та медіа-проєктах.

ПРОТОКОЛ ПЕРЕДАЧІ ДАНИХ, МУЛЬТИМЕДІЙНА ТРАНСЛЯЦІЯ, TCP, UDP, SRT, HLS, WEBRTC, QUIC, QOS, QOE.

ABSTRACT

The purpose of the work is to analyze data transmission protocols in electronic communication networks to ensure efficient real-time multimedia broadcasting, identify their strengths and weaknesses, and determine the optimal usage based on the broadcast scenario.

The thesis investigates the core characteristics of protocols such as TCP, UDP, SRT, WebRTC, HLS, MPEG-DASH, and QUIC. It presents a comparative analysis in terms of latency, adaptiveness, scalability, reliability, and resource efficiency. Evaluation criteria such as packet loss, jitter, playback quality, compatibility, and transmission security are also explored. The paper offers practical recommendations for selecting the appropriate protocol for various broadcast types (VoD, live, interactive), depending on network constraints and technical requirements. Special attention is given to the impact of 5G, edge computing, and AI-driven adaptation, justifying the relevance of hybrid architectures for high-quality streaming in esports, education, and media applications.

DATA TRANSMISSION PROTOCOL, MULTIMEDIA STREAMING, TCP, UDP, SRT, HLS, WEBRTC, QUIC, QOS, QOE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ABR	Adaptive Bitrate Streaming — адаптивна передача потоку з автоматичним підбором якості
ACK	Acknowledgment — підтвердження прийому пакета
API	Application Programming Interface — інтерфейс прикладного програмування
ARP	Address Resolution Protocol — протокол визначення адреси
BGP	Border Gateway Protocol – протокол граничного шлюзу
CDN	Content Delivery Network — мережа доставки контенту
CMAF	Common Media Application Format — уніфікований формат HLS та MPEG-DASH
CSMA/CD	Carrier Sense Multiple Access with Collision Detection — доступ до каналу з виявленням колізій
DNS	Domain Name System — система доменних імен
DSCP	Differentiated Services Code Point — мітка QoS у IP-пакетах
FEC	Forward Error Correction — пряма корекція помилок
FTP	File Transfer Protocol — протокол передачі файлів
GNS3	Graphical Network Simulator — система емуляції мереж
HLS	HTTP Live Streaming — протокол сегментованого стрімінгу
HTTP	HyperText Transfer Protocol — протокол гіпертексту
ICMP	Internet Control Message Protocol — протокол контролю мережевих повідомлень
IoT	Internet of Things — інтернет речей
IP	Internet Protocol — міжмережевий протокол
IPv4	Internet Protocol version 4 — протокол IP четвертої версії

IPv6	Internet Protocol version 6 — протокол IP шостої версії
ISP	Internet Service Provider — провайдер інтернет-послуг
Jitter	Коливання часу між надходженням пакетів
LAN	Local Area Network — локальна мережа
Latency	Затримка в мережі
MAC	Media Access Control — управління доступом до середовища передачі
MEC	Multi-access Edge Computing — обчислення на краю мережі
MPEG-DASH	Dynamic Adaptive Streaming over HTTP — адаптивна передача відео через HTTP
NAT	Network Address Translation — трансляція IP-адрес
OSI	Open Systems Interconnection — модель взаємодії відкритих систем
OSPF	Open Shortest Path First — протокол пошуку найкоротшого шляху
QoE	Quality of Experience — якість досвіду користувача
QoS	Quality of Service — якість обслуговування
QUIC	Quick UDP Internet Connections — швидкий транспортний протокол на базі UDP
RAM	Random Access Memory — оперативна пам’ять
RFC	Request for Comments — стандартизаційний документ IETF
RIP	Routing Information Protocol — протокол маршрутизації
RTP	Real-time Transport Protocol — протокол реального часу
RTMP	Real-Time Messaging Protocol — протокол потокового відео
RTSP	Real-Time Streaming Protocol — протокол керування медіапотоками

SRT	Secure Reliable Transport — безпечний протокол відеотрансляції з компенсацією втрат
TCP	Transmission Control Protocol — транспортний протокол з контролем потоку
UDP	User Datagram Protocol — простий транспортний протокол без підтвердження
VoD	Video on Demand — відео на запит
VLAN	Virtual LAN — віртуальна локальна мережа
VPN	Virtual Private Network — віртуальна приватна мережа
WAN	Wide Area Network — глобальна мережа
WebRTC	Web Real-Time Communication — реального часу веб-комунікації
XML	Extensible Markup Language — розширювана мова розмітки

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації всіх сфер суспільного життя питання ефективної передачі мультимедійних даних набуває особливої актуальності. Застосування мереж електронних комунікацій стає основою для прямих трансляцій, онлайн-конференцій, відеомоніторингу, електронного телебачення та інших сервісів, які щодня використовують величезні обсяги інформації. Це зумовлено зростанням кількості пристроїв, підключених до мереж, збільшенням трафіку та підвищенням вимог до якості зв’язку, що створює необхідність у постійному вдосконаленні протоколів передачі даних та оптимізації їх роботи у реальному часі.

Мета роботи полягає у комплексному аналізі існуючих протоколів передачі даних в умовах мультимедійного мовлення прямих трансляцій, визначенні їх сильних і слабких сторін з метою формування практичних рекомендацій щодо вдосконалення підходів до організації передачі мультимедійних потоків у сучасних мережах. Для досягнення поставленої мети дослідження включає кілька завдань: здійснення систематичного огляду сучасних протоколів, аналіз технічних характеристик та ефективності роботи кожного з них у контексті прямих трансляцій; проведення компаративного аналізу різних підходів до передачі даних, що використовуються в мережах електронних комунікацій; розробку критеріїв оцінки ефективності протоколів; моделювання сценаріїв роботи мережі за участю мультимедійних потоків, а також виявлення оптимальних технологій для специфічних умов експлуатації.

Об’єктом дослідження є сучасні електронні комунікаційні мережі, які забезпечують передачу мультимедійних даних у режимі реального часу, тоді як предметом дослідження виступають протоколи передачі даних, їх функціональні можливості, технічні параметри та методи оптимізації для забезпечення високої якості мультимедійного мовлення. Використані методи дослідження охоплюють аналітичний огляд наукової літератури,

порівняльний аналіз технічних характеристик протоколів, математичне моделювання процесів передачі даних у мережах, експериментальні дослідження в лабораторних умовах, а також використання сучасного програмного забезпечення для симуляції мережевих потоків.

Структура роботи розділена на кілька логічно послідовних частин: вступ, у якому викладено актуальність теми, мету, завдання, об’єкт і предмет дослідження, методи дослідження та загальну схему роботи; розділи, присвячені теоретичним основам протоколів передачі даних, детальному аналізу їх технічних характеристик і функціональних можливостей, а також компаративному аналізу різних підходів до організації мережевих потоків для мультимедійного мовлення; висновки, в яких узагальнюються отримані результати, формулюються рекомендації щодо подальших досліджень і впровадження оптимізаційних заходів, а також розглядаються перспективи розвитку технологій передачі мультимедійного контенту. Загалом, дана робота має на меті не лише провести детальний аналіз існуючих рішень, але й окреслити напрямки для подальших досліджень, що дозволять підвищити ефективність роботи мереж електронних комунікацій та забезпечити високу якість мультимедійного мовлення в умовах постійно зростаючих вимог до швидкості та надійності передачі даних.

1 ТЕОРЕТИЧНІ ОСНОВИ МУЛЬТИМЕДІЙНОГО МОВЛЕННЯ

1.1 Поняття мультимедіа та прямих трансляцій: характеристики і відмінності від інших типів контенту

Мультимедіа — це форма подання інформації, яка поєднує кілька типів медіа, зокрема текст, графіку, аудіо, відео, анімацію та інтерактивні елементи. Основна мета мультимедіа — забезпечити максимально наочну, зручну та ефективну комунікацію з користувачем, використовуючи можливості кількох сенсорних каналів одночасно. Сфера застосування мультимедійних технологій охоплює навчання, рекламу, мистецтво, ігрову індустрію, журналістику, телебачення, а також інформаційні сервіси в Інтернеті.

Пряма трансляція — це специфічна форма мультимедійного контенту, яка передбачає передачу аудіовізуального потоку в режимі реального часу, без попереднього запису або редагування. Найпоширенішими платформами для таких трансляцій є стрімінгові сервіси (YouTube Live, Twitch, Facebook Live), професійні рішення для телемовлення, а також спеціалізовані програмні комплекси, які використовуються в кіберспорті, новинах, подієвому мовленні тощо. Пряма трансляція має характерні особливості: мінімальна затримка між подією та її відображенням на пристрої користувача, висока вимогливість до стабільного та швидкого з’єднання, необхідність забезпечення якості відео- та аудіопотоку в реальному часі, а також підтримка масштабованості для одночасного перегляду великою кількістю користувачів.

Ключовою відмінністю прямих трансляцій від інших типів мультимедійного контенту (наприклад, записаного відео, подкастів або інтерактивних курсів) є часовий чинник — контент не існує заздалегідь, а генерується та доставляється до кінцевого користувача в режимі "тут і зараз". Це створює низку технічних викликів: потребу в мінімізації затримки (latency), адаптації до змін пропускної здатності мережі, динамічного кодування відео, балансування навантаження на сервери тощо. Інші види контенту можуть бути попередньо оброблені, оптимізовані, стиснуті та збережені на локальних чи

хмарних сховищах, що дозволяє уникнути багатьох технічних труднощів, притаманних прямому мовленню.

Таким чином, мультимедіа охоплює широкий спектр інформаційних форматів, тоді як пряма трансляція є найбільш вимогливою формою мультимедійного контенту з точки зору організації технічної інфраструктури, забезпечення якості передачі даних та надійності мережевого середовища.

1.2 Вимоги до якості передачі потокового контенту

Якість сприйняття потокового мультимедійного контенту визначається сукупністю параметрів мережевої взаємодії та параметрів програвача, які впливають на безперервність відтворення та затримку між подією та її відображенням. Найважливішими з них є затримка (latency), механізми буферизації, політики Quality of Service (QoS) і здатність системи до адаптивного потокового передавання (adaptive bitrate streaming).

Затримка визначає інтервал між моментом захоплення кадру на джерелі та відображенням його у глядача. Сучасні протоколи демонструють значний розкид затримок: WebRTC здатен забезпечувати субсекундні показники, іноді менше 500 мс, RTMP/RTSP утримують затримку в межах 2–5 с, а класичні HLS/DASH—до 10–45 с без їхніх low-latency розширень. Впровадження Low-Latency HLS або CMAF-DASH дозволяє знизити цей показник до 2–5 с, що особливо важливо для інтерактивних подій, спортивних трансляцій та івентів із прямою взаємодією з аудиторією.

Буферизація полягає в попередньому накопиченні певного обсягу даних у плеєрі перед відтворенням. Типовий початковий буфер становить 2–10 с відео, що зменшує ризик переривань через коливання пропускної здатності мережі. Проте надмірна буферизація підвищує початкову затримку, а її недостатність викликає часті паузи. Сучасні плеєри використовують динамічне регулювання розміру буфера: при виявленні «заїдань» автоматично підвищують обсяг кешу, а під час стабільної передачі — зменшують, щоб

мінімізувати затримку. Крім того, застосування CDN-рішень дозволяє кешувати сегменти ближче до користувачів, додатково знижуючи ймовірність буферизаційних пауз.

Quality of Service (QoS) на рівні мережі гарантує пріоритетність мультимедійного трафіку над іншими даними. На обладнанні Cisco для цього налаштовують клас-мапи і політики, маркують відео- та аудіопакети DSCP-позначеннями (зокрема EF — Expedited Forwarding) і застосовують чергу priority queue для забезпечення мінімальних втрат і джиттера. Такі політики дозволяють утримувати якість трансляції навіть у періоди пікових навантажень, запобігаючи «зависанням» та «пікселізації» відео.

Адаптивне потокове передавання (Adaptive Bitrate Streaming) забезпечує максимальну якість зображення за поточних умов каналу, динамічно перемикаючись між кількома бітрейтами. Протоколи HLS та MPEG-DASH розбивають відео на сегменти довжиною 2–10 с, кожен із яких закодовано в кількох якостях. Клієнтський плеєр аналізує поточну швидкість мережі й обирає оптимальний сегмент, що дозволяє уникати буферизацій при зниженні пропускної здатності та значно покращує якість на швидких каналах. Використання коротких сегментів (1–2 с) у Low-Latency CMAF/DASH та LL-HLS забезпечує більш оперативне реагування на зміни пропускної здатності без втрати адаптивності .

Таким чином, для успішної організації прямих трансляцій необхідно зводити до мінімуму затримку, оптимізувати буферизацію, конфігурувати QoS-пріоритети та впроваджувати адаптивні алгоритми передавання. Лише поєднання всіх цих підходів гарантує стабільний, якісний та практично безперервний досвід для кінцевого глядача.

1.3 Огляд архітектури мереж електронних комунікацій

Архітектура мереж електронних комунікацій визначає, як саме пристрої об’єднуються й взаємодіють для обміну даними. Від вибору типу мережі

залежить пропускна здатність і затримки, а референтні моделі рівнів допомагають стандартизувати протоколи та сервіси. Топології ж описують фізичну чи логічну схему підключень, що напряму впливає на надійність, масштабованість і вартість розгортання.

Локальна мережа (LAN) об’єднує пристрої в межах одного обмеженого простору — будівлі, офісу або кампусу. LAN забезпечує високу швидкість передачі даних — від 100 Мбіт/с до 10 Гбіт/с і більше в сучасних реалізаціях — та мінімальні затримки завдяки дротовим (Ethernet, IEEE 802.3) або бездротовим (Wi-Fi, IEEE 802.11) технологіям. Здебільшого LAN складається з комутаторів, маршрутизаторів, точок доступу та структурованої кабельної розводки Cat5e/6/6a, що дає змогу централізовано керувати трафіком і забезпечувати якісну доставку мультимедійних потоків у межах організації. Окремо виділяють бездротові LAN (WLAN) з гнучкістю розгортання там, де не можна прокладати кабелі; вони працюють у діапазонах 2,4 ГГц, 5 ГГц і 6 ГГц (Wi-Fi 6/6E) та підтримують швидкість до гігабіт у секунду за сприятливих умов. Логічно в межах LAN часто створюють віртуальні локальні мережі (VLAN) для сегментації трафіку за відділами чи функціональними групами без змін фізичної інфраструктури.

Міські мережі (MAN) розширюють LAN на рівень міста або кампусу, з’єднуючи віддалені офіси або корпуси університету за допомогою волоконно-оптичних магістралей, радіоканалів або орендованих каналів цифрового зв’язку. Пропускна здатність MAN сягає сотень мегабіт на секунду й більше, а зони покриття — до десятків кілометрів. Для гарантування якості обслуговування трафіку в MAN часто застосовують технології Metro Ethernet із підтримкою MPLS та SD-MAN, що забезпечують пріоритезацію мультимедіа, балансування навантаження й швидке відновлення після відмови ділянки. Міські мережі використовують кільцеві чи зіркові архітектури з резервуванням каналів, щоб уникнути простоїв і забезпечити високу готовність сервісів.

Глобальні мережі (WAN) з’єднують LAN і MAN на національному та міжконтинентальному рівнях через орендовані лінії (T1/E1), MPLS-сервіси, супутникові канали, VPN через Інтернет і SD-WAN, що використовує політики на основі програмно-визначуваних мереж. WAN охоплюють відстані від сотень до тисяч кілометрів, мають затримки від десятків до сотень мілісекунд і швидкості від сотень кілобіт до сотень мегабіт на секунду залежно від технології. Для маршрутизації в WAN застосовують протоколи BGP (для обміну маршрутами між автономними системами) і OSPF (для внутрішньомережевої маршрутизації), що забезпечують вибір оптимального шляху та автоматичне перенаправлення трафіку при збоях.

Персональні мережі (PAN), наприклад Bluetooth (IEEE 802.15.1), покривають близько 10 м і створюють низькошвидкісні енергоефективні з’єднання для периферії — гарнітур, сенсорів, носимих пристроїв . PAN широко використовуються в сценаріях Інтернету речей (IoT) для обміну невеликими обсягами даних із мінімальною затримкою й енергоспоживанням.

Взаємодію мережевого обладнання, сервісів і користувацьких застосунків стандартизують за допомогою референтних моделей рівнів.

Модель OSI (Open Systems Interconnection) описує сім шарів із чіткими функціями, починаючи від фізичного, що відповідає за генерування і прийом електричних чи оптичних сигналів по середовищу, до прикладного, який реалізує сервіси рівня HTTP, FTP, DNS, SMTP тощо. Фізичний шар визначає параметри кабелів, конекторів, напругу, частоту та режим роботи (simplex/duplex); канальний шар відповідає за інкапсуляцію фреймів і корекцію помилок (Ethernet MAC, Wi-Fi LLC); мережевий — маршрутизує IP-пакети й реалізує адресацію (IP, ICMP, IGMP); транспортний — забезпечує надійність та контроль потоку (TCP) або швидкість і мінімальні накладні витрати (UDP); сесійний — встановлює та керує діалогами між додатками (RPC, NetBIOS); представлення — перетворює формати даних і виконує шифрування/дешифрування (TLS/SSL, JPEG, MPEG); прикладний — інтерпретує запити користувача й надає доступ до ресурсів мережі.

Модель TCP/IP (Internet Protocol Suite) спрощує ієрархію до чотирьох шарів: канального (Link), мережевого (Internet), транспортного (Transport) та прикладного (Application), що відповідає реальним протоколам IP, ICMP, TCP, UDP та HTTP/FTP/SMTP . У TCP/IP канальний шар включає функції OSI фізичного й канального; інтернет-шар — маршрутизацію IP; транспортний — контроль доставки й відновлення; прикладний — реалізацію сервісів користувача.

Топологія мережі описує фізичне або логічне розташування з'єднань між вузлами і впливає на продуктивність, стійкість і масштабованість.

Шинна топологія об'єднує всі вузли спільним полудуплексним кабелем (Bus), по якому дані передаються до всіх станцій одночасно. Переваги — проста й недорога розводка, одна лінія з'єднання; недоліки — обрив шини призводить до зупинки всієї мережі, спільний колізійний домен, обмежена довжина сегмента і пропускна здатність.

Зіркоподібна топологія з'єднує кожен вузол окремим кабелем із центральним комутатором або концентратором. Ця схема підвищує надійність — збій одного кінцевого з'єднання не впливає на інші; водночас вихід з ладу центрального елемента паралізує мережу. Додавання нових пристроїв відбувається без перерви роботи, що зручно для масштабування.

Кільцева топологія організовує вузли в замкнений контур, де кожен передає дані сусідові по маркер-процесу або за допомогою токенів (Token Ring, FDDI). Така схема дозволяє уникнути колізій і точно розрахувати час доставки; проте вихід з ладу одного вузла або лінії без резервного каналу призведе до розриву ланцюга.

Сіткова топологія (Mesh) передбачає численні з'єднання між вузлами, що створює множинні шляхи передачі й забезпечує найвищу відмовостійкість і мінімальні затримки. Ця архітектура часто використовуються в індустріальних чи критичних системах, але вимагає значних витрат на кабелі або радіомодулі та комплексного адміністрування.

Гібридні топології поєднують базові схеми для досягнення балансу між надійністю, продуктивністю й вартістю. Наприклад, древовидна (Tree або Star-Bus) структура об’єднує кілька зіркоподібних сегментів через шину; часто реалізується в корпоративних мережах, де кінцеві відділи підключені зіркою до магістралі.

Вибір архітектурних компонентів — типу мережі, мережевої моделі та топології — визначає ефективність доставки мультимедійних даних у режимі реального часу. Для прямих трансляцій важливі низькі затримки та висока надійність, тому найчастіше застосовують комбінації LAN із резервованими каналами, MPLS-WAN із QoS-політиками та гібридні топології з елементами mesh, що гарантують мінімальні простої та відмови компонентів.

1.4. Види протоколів передачі даних та їх класифікація

Протоколи передачі даних — це набір правил, що регламентують формат, порядок і методи обміну інформацією між пристроями в електронних комунікаційних мережах. Вони є основою функціонування як глобального Інтернету, так і локальних мереж, дозволяючи різноманітним пристроям ефективно обмінюватися даними, незалежно від їх апаратного чи програмного забезпечення. Протоколи працюють на різних рівнях моделі OSI або TCP/IP, забезпечуючи все — від фізичної передачі бітів до надійного транспортування відео в реальному часі.

На фізичному та канальному рівнях знаходяться базові протоколи, що регламентують, як дані кодуються і передаються по конкретному середовищу: дроту, оптичному волокну чи бездротовому каналу. Ethernet (IEEE 802.3) — це найбільш розповсюджений стандарт для локальних мереж, який використовує MAC-адресацію для доставки кадрів. У бездротових мережах застосовуються протоколи на кшталт Wi-Fi (IEEE 802.11), які поєднують фізичний і канальний рівні та дозволяють передачу даних у середовищах з високим рівнем перешкод. Інші приклади — PPP (Point-to-Point Protocol), що

встановлює прямі з’єднання між двома вузлами, та HDLC для серійних ліній передачі.

На мережевому рівні основну роль відіграє IP (Internet Protocol), що відповідає за маршрутизацію та адресацію пакетів. IPv4 залишається найпоширенішою версією, хоча його наступник IPv6 поступово впроваджується, завдяки ширшому діапазону адрес. Супутні протоколи, як ARP (Address Resolution Protocol), допомагають перетворювати IP-адреси в MAC-адреси, а ICMP (Internet Control Message Protocol) забезпечує діагностику та повідомлення про помилки (наприклад, у команді ping). У складніших архітектурах використовуються MPLS (Multiprotocol Label Switching) — протокол для маршрутизації трафіку за мітками, що дозволяє забезпечити QoS для потокових даних.

На транспортному рівні TCP (Transmission Control Protocol) і UDP (User Datagram Protocol) є двома фундаментальними протоколами. TCP орієнтований на з’єднання, тобто гарантує доставку даних у правильному порядку та без втрат, що критично для передачі файлів, електронної пошти чи веб-сторінок. UDP, навпаки, надає пріоритет швидкості, нехтуючи перевірками, що робить його зручним для потокового відео та голосових сервісів. QUIC, новий протокол, розроблений Google, об’єднує переваги UDP з надійністю TCP і безпечним з’єднанням TLS, забезпечуючи меншу затримку під час встановлення з’єднання.

На рівні прикладних сервісів працюють численні протоколи. HTTP/HTTPS забезпечують передачу веб-контенту, FTP/SFTP/TFTP — обмін файлами, SMTP/POP3/IMAP — роботу електронної пошти. DNS (Domain Name System) виконує резолюцію доменних імен у IP-адреси, а DHCP (Dynamic Host Configuration Protocol) автоматично надає IP-адреси пристроям у мережі. Усе це дозволяє користувачам взаємодіяти з Інтернетом та внутрішніми мережами без складного ручного налаштування.

Щодо класифікації, першим критерієм є тип з’єднання. Протоколи з’єднання (TCP, FTP, SCTP) встановлюють стійкий канал між пристроями,

забезпечуючи підтвердження отримання, повторну передачу втрат та контроль помилок. Безз'єднувальні протоколи (UDP, ICMP, TFTP) передають пакети окремо, без попередньої сесії, і не гарантують доставку. Перші підходять для критично важливих даних, другі — для сценаріїв реального часу.

Другий критерій — рівень моделі OSI. Фізичний і канальний рівень включають Ethernet, Wi-Fi, Bluetooth. Мережевий — IP, ICMP, ARP, MPLS. Транспортний — TCP, UDP, QUIC, SCTP. Прикладний — HTTP, FTP, DNS, SMTP, RTP, SIP, SNMP тощо. Кожен рівень виконує свої функції і забезпечує потрібні сервіси для вищих рівнів.

За функціональністю можна виділити кілька груп протоколів. Потокові мультимедіа-протоколи забезпечують передачу аудіо- і відеопотоків. Серед них — RTP (Real-time Transport Protocol), що додає часові мітки та номери кадрів; RTSP (Real-Time Streaming Protocol) дозволяє керування потоком (відтворення, пауза, перемотування); SRT (Secure Reliable Transport) забезпечує захищену передачу відео з мінімальними затримками. Ці протоколи відіграють ключову роль у реалізації онлайн-трансляцій, відеоконференцій та IPTV.

Протоколи обміну повідомленнями часто використовуються в IoT. MQTT (Message Queuing Telemetry Transport) — легкий протокол публікації/підписки, оптимізований для пристроїв з обмеженими ресурсами. AMQP (Advanced Message Queuing Protocol) — більш складний і підтримує черги, підтвердження, маршрутизацію, що важливо для enterprise-рішень.

Протоколи управління мережею включають SNMP (Simple Network Management Protocol), що дозволяє адміністраторам моніторити і керувати обладнанням через агенти на пристроях. NetFlow і sFlow надають інформацію про структуру трафіку, допомагаючи оптимізувати мережеву архітектуру та виявляти аномалії.

Окремо виділяють протоколи маршрутизації. Внутрішньомережеві (IGP) — OSPF (Open Shortest Path First) використовує алгоритм Дейкстри для побудови найкоротших маршрутів у межах автономної системи. EIGRP —

гібридний протокол від Cisco. Зовнішній — BGP (Border Gateway Protocol), який керує маршрутизацією між автономними системами та забезпечує стабільність глобального Інтернету.

Також розрізняють протоколи за типом адресації: унікаст (один-в-одного, як у TCP чи HTTP), мультикаст (один-багатьом, як у IPTV або онлайн-лекціях через IGMP), бродкаст (один-усім у мережевому сегменті, як у DHCP Discover чи ARP-запитах). Вибір типу адресації залежить від поставлених цілей та архітектури мережі.

Загалом, кожен протокол має свої переваги, недоліки та специфічні сценарії використання. Для побудови ефективних мультимедійних систем важливо грамотно обирати протоколи залежно від вимог до затримки, надійності, масштабованості, пропускної здатності й безпеки. Комплексне розуміння класифікації та характеристик протоколів є основою для проектування сучасних комунікаційних мереж та реалізації якісного мультимедійного мовлення в реальному часі.

1.5 Роль протоколів у забезпеченні якості сервісу (QoS)

Якість обслуговування (QoS) визначає здатність мережі гарантувати певні характеристики передачі — мінімальні затримки, обмежений рівень втрат, передбачувану пропускну здатність та контроль джиттера. Саме протоколи забезпечують реалізацію QoS-механізмів на різних рівнях мережі, від відокремлення трафіку й маркування пакетів до пріоритезації та ресурсного резервування, що є критично важливим для мультимедійних трансляцій, VoIP, фінансових транзакцій або IoT-сервісів.

По-перше, архітектури інтегрованих сервісів (IntServ) та протокол RSVP дозволяють виділяти ресурси перетинами мережі для конкретних потоків. RSVP (Resource Reservation Protocol) встановлює перед початком передачі резервацію пропускної здатності й підтримує її протягом сеансу, гарантуючи надійну доставку аудіо- та відеоданих із заданими затримками й втратами. Цей

підхід “per-flow” підходить для невеликих груп клієнтів, але погано масштабується в глобальних мережах через високі обчислювальні та контролюючі навантаження на маршрутизатори.

У відповідь на обмеження IntServ, архітектура розрізнених сервісів (DiffServ) використовує маркування пакетів за допомогою DSCP (Differentiated Services Code Point) у заголовку IP. Класи трафіку (кристалізовані 6-бітним DSCP) групують потоки з подібними вимогами до затримки та пропускної здатності, надаючи пріоритет критичним мультимедійним даним, наприклад, VoIP або відеоконференціям, і best-effort-обслуговування менш важливому трафіку. DiffServ легко масштабується у великих мережах, оскільки не потребує збереження стану кожного потоку, лише обробки груп трафіку.

На мережевому рівні MPLS Traffic Engineering (MPLS-TE) у поєднанні з RSVP-TE створює керовані тунелі (LSP) із гарантованою пропускною здатністю. RSVP-TE сигналізує шлях, резервуючи ресурси на кожному маршрутизаторі — це дозволяє детально балансувати навантаження та мінімізувати затримки для критичного трафіку. MPLS-TE активно використовується у магістральних мережах операторів зв’язку, де важлива відмовостійкість і передбачуване QoS.

Ключові мережеві елементи реалізують політики обробки трафіку через traffic policing та traffic shaping. Поліси обмежують швидкість вихідного трафіку, відкидаючи або позначаючи надлишкові пакети, тоді як форсоване формування (shaping) накопичує надлишок у буфері й відправляє згладженими хвилями, запобігаючи різким стрибкам навантаження. Окрім Cisco IOS XE, подібні механізми доступні в Juniper Junos та інших платформах, де конфігуруються commit-rate, burst-size та інші параметри.

Для пріоритезації трафіку застосовують сортування черг (queuing): WRED (Weighted Random Early Detection) відкидає пакети за раннім сигналом про затори, CBWFQ (Class-Based Weighted Fair Queuing) гарантує мінімальну пропускну здатність кожному класу, а LLQ (Low Latency Queuing) виділяє

окрему чергу з найвищим пріоритетом для критичних потоків, таких як голос і відео. Це забезпечує важливим застосункам низький джиттер та затримку.

Розширені можливості QoS шукають на транспортному рівні через TCP-конгест-контроль (AIMD, slow-start, congestion avoidance) та Explicit Congestion Notification (ECN), який дозволяє маршрутизаторам маркувати пакети замість їх відкидання, передаючи сигнал про завантаження без втрат пакета. ECN-маркування стежить через два біти в полі DSCP, підвищуючи ефективність та зменшуючи таймаути під час надмірного навантаження.

У локальних мережах для відокремлення аудіотрафіку від даних IEEE 802.1p/CoS у VLAN використовує 3-бітову відмітку пріоритету в заголовку Ethernet, що дає сім рівнів пріоритетів для забезпечення негайної обробки важливих кадрів на комутаторах. Подібні механізми доступні і в мобільних мережах у вигляді IEEE 802.11e EDCA, де голосові пакети отримують коротші інтервали доступу до середовища.

На прикладному рівні протоколи потокової передачі даних мають власні QoS-механізми: RTP додає часові мітки та номери пакетів для корекції джиттера та синхронізації, а RTCP моніторить статистику доставки й повідомляє відправника про якість, дозволяючи динамічно коригувати бітрейт. У стеках WebRTC застосовуються SCTP-канали над DTLS-шифруванням із adaptative bitrate (ABR) і congestion control, що гарантує оптимальну якість навіть у змінних умовах мережі.

Нарешті, моніторинг і управління QoS здійснюються через SNMP, NetFlow, sFlow та інші телеметричні протоколи. SNMP-агенти на маршрутизаторах та комутаторах експонують показники затримок, втрат, заповненості черг і реєструють події політики. NetFlow аналізує характеристики трафіку по потоках, допомагаючи коригувати політики QoS у реальному часі.

Таким чином, сукупність протоколів та механізмів — резервування ресурсів (RSVP/IntServ), маркування та клас-базоване обслуговування (DiffServ/DSCP, 802.1p), тунелювання й маршрутизація з резервуванням

(MPLS-TE), обмеження та формування трафіку (policing/shaping), сортування черг і раннє виявлення заторів (WRED, LLQ), транспортні конґест-контролі й ECN, а також моніторинг і зворотний зв’язок (RTCP, SNMP, NetFlow) — утворюють комплексний підхід до QoS. Кожен рівень моделі та кожен тип протоколу вносять свій внесок у забезпечення стабільності, надійності та передбачуваності мережевого сервісу, що є запорукою високої якості прямих мультимедійних трансляцій і інших критичних застосунків.

1.6 Висновки з розділу 1

У першому розділі було проведено системний аналіз архітектури електронних комунікаційних мереж, що забезпечують мультимедійне мовлення у режимі реального часу. Розглянуто фундаментальні принципи побудови мереж передачі даних, охарактеризовано особливості протоколів транспортного рівня, а також технологічні вимоги до середовищ мовлення потокового відео.

Окрема увага приділена сучасним моделям доставки мультимедіа — як на основі централізованих серверів, так і із застосуванням CDN-інфраструктур. Підкреслено зростаючу роль протоколів нового покоління в умовах масштабування трафіку відеотрансляцій та інтеграції з хмарними платформами.

У процесі аналізу було визначено ключові параметри якості передачі: затримка, джиттер, втрата пакетів, надійність доставки та вимоги до адаптивності у мережах з різною пропускною здатністю. Зокрема, підкреслено значення протоколів з підтримкою FEC, congestion control та low-latency delivery у побудові ефективної мультимедійної інфраструктури.

Таким чином, розділ створює концептуальну основу для подальшого порівняння конкретних протоколів (TCP, UDP, SRT, HLS, QUIC, WebRTC) у наступних розділах, а також визначає основні вимоги до систем

мультимедійної трансляції, які повинні враховуватись при розробці і виборі рішень у сфері live streaming.

2 АНАЛІЗ ПРОТОКОЛІВ ПЕРЕДАЧІ ДАНИХ ДЛЯ ПРЯМОГО МОВЛЕННЯ

2.1 Протоколи загального призначення: TCP та UDP

2.1.1 Протокол TCP: структура та встановлення з’єднання

TCP — складний транспортний протокол, що забезпечує надійну передачу потокових даних у мережах IP. Базовий заголовок TCP займає 20 байт (з можливістю розширення опціями до 60 байт) і містить порти джерела та призначення, порядковий номер сегмента і номер підтвердження, поле Data Offset для вказівки довжини заголовка, зарезервовані біти, прапорці (SYN, ACK, PSH, RST, FIN тощо) і допоміжні поля: розмір вікна, контрольну суму та указівник термінових даних. Прапорці керують стадіями сеансу та сигналізують про особливі умови передачі (наприклад, SYN ініціює з’єднання, ACK підтверджує прийом, FIN означає завершення передачі), а поле Window Size реалізує механізм контролю потоку, повідомляючи, скільки байт приймач здатен прийняти без підтвердження. Контрольна сума, обчислена за допомогою псевдо-заголовка IP, перевіряє цілісність сегмента й допомагає виявляти пошкодження під час передачі.

Для розширення функціональності TCP використовуються опції заголовка, які можуть складати до 40 байт. Найважливіші серед них — MSS (Maximum Segment Size), що визначає максимальний розмір сегмента без фрагментації; Window Scale, що дозволяє збільшити розмір приймального вікна понад 65 535 байт (особливо актуально при великих RTT); SACK (Selective Acknowledgment), який дає змогу повідомляти про отримання окремих блоків даних поза загальною послідовністю і значно покращує відновлення при втраті; а також Timestamps, що додають часові мітки до сегментів для точного вимірювання RTT і уникнення плутанини з повторними ACK. Інші опції (NOP, Padding та ін.) узгоджують вирівнювання заголовка та підтримують додаткові функціональні можливості.

Процедура встановлення з’єднання в TCP побудована на триетапному рукостисканні: клієнт надсилає сегмент із прапорцем SYN і початковим ISN,

сервер у відповідь надсилає SYN+ACK (ACK = ISN клієнта + 1 та власний ISN), а клієнт завершує handshake, відправивши ACK (ACK = ISN сервера + 1). Після цього з’єднання перебуває в стані ESTABLISHED і готове до обміну даними. Надійність передачі забезпечується нумерацією байтів і механізмом підтверджень ACK: якщо відправник не отримує ACK у встановлений час, сегмент вважається втраченим і ретрансльований. Завершення сеансу відбувається через обмін сегментами з прапорцем FIN у чотири кроки (FIN → ACK → FIN → ACK), що гарантує впорядковане закриття. У разі екстреного скидання використовується прапорець RST, який негайно розриває з’єднання без додаткових узгоджень.

2.1.2 Управління потоком і заторами у TCP

Однією з ключових рис TCP є механізми контролю, що забезпечують стабільну роботу в різних умовах. Управління потоком (Flow Control) реалізується через поле Window Size: отримувач повідомляє, скільки байт буфера вільно, а відправник не надсилає більше непідтверджених даних, ніж дозволяє вікно. Якщо вікно рівне нулю, передача призупиняється до оновлення, а таймер Persist періодично відправляє Zero Window Probe, щоб отримати новий розмір вікна. Це гарантує, що сеанс не зависне назавжди, і передача поновиться, щойно з’явиться вільний буфер.

Управління заторами (Congestion Control) захищає мережу від перевантаження. За класичним RFC 5681, нове з’єднання або з’єднання після тайм-ауту починає із фази Slow Start — щойно надсилається один MSS сегмент за кожен отриманий ACK, що призводить до експоненційного зростання cwnd (congestion window) до порогу ssthresh. Після досягнення ssthresh переходять до Congestion Avoidance, де cwnd збільшується вже лінійно. За поодинокої втрати (поява дублікат-ACK) запускається Fast Retransmit (невідкладна ретрансмісія втраченого сегмента) і Fast Recovery, коли cwnd тимчасово зменшується приблизно вдвічі і потім зростає поступово, замість повного

повернення до Slow Start. Якщо ж втрати відбуваються без жодних ACK, відновлення починається із Slow Start із cwnd = 1 MSS та ssthresh удвічі меншим від попереднього. Завдяки цьому TCP «намацує» оптимальну пропускну здатність, швидко реагує на незначні втрати, але стримує передачу при ознаках сильного затору.

Сучасні вдосконалення додають адаптивність: CUBIC (за замовчуванням у Linux) збільшує cwnd за кубічною функцією часу, що краще підходить для високошвидкісних лінків; BBR оцінює реальну пропускну здатність та мінімальний RTT, подаючи дані на межі можливостей мережі, не переповнюючи черги. Інші алгоритми (Hybla, Vegas, PCC тощо) оптимізовані під різні сценарії. Опції заголовка, такі як SACK, Window Scale та Timestamps, також прискорюють виявлення втрат і настроювання таймаутів, що робить TCP стійким і адаптивним навіть за високих затримок чи значного завантаження мережі. Завдяки цим вбудованим механізмам надійності та контролю TCP залишається домінуючим протоколом транспортного рівня.

2.1.3 Протокол UDP: структура та застосування

На відміну від TCP, UDP не встановлює з’єднання і не контролює доставку – він просто надсилає незалежні дейтаграми без жодного попереднього узгодження. Заголовок UDP займає лише 8 байт і містить чотири поля по 16 біт: порт джерела, порт призначення, довжину пакета й контрольну суму. Контрольна сума може охоплювати весь пакет (заголовок + дані) або, якщо відправник вкаже нулі, взагалі ігноруватися, що ще більше зменшує затримки. Відсутність механізмів підтвердження, повторних передач чи впорядкування означає, що відповідальність за виявлення втрачених пакета або сортування отриманих байтів покладається на вищі рівні стеку. Проте саме завдяки такій простоті UDP має мінімальні накладні витрати і забезпечує неймовірно швидкий старт передачі — достатньо одного пакета, щоб розпочати обмін даними.

Простота UDP робить його ідеальним для тих сценаріїв, де затримка важливіша за стовідсоткову надійність. Наприклад, у мультимедіа-додатках реального часу (VoIP, відеоконференції й онлайн-трансляції) кожен мілісекунд—на вагу золота: метод «відправив і забув», який пропонує UDP, дозволяє починати передавання практично миттєво, а невеликий заголовок (8 байт проти щонайменше 20 байт у TCP) суттєво зменшує протокольний оверхед при передачі великих потоків аудіо чи відео. Якщо якийсь пакет втрачено, додаток може просто пропустити фрейм або застосувати механізми згладжування; додаткове впорядкування й нумерацію дейтаграм часто виконують RTP/RTCP, що додає часові мітки та звіти про якість каналу. Подібним чином у середовищі IoT, де багато пристроїв мають обмежену пам’ять і енергію, UDP дозволяє передавати короткі телеметричні повідомлення з мінімальними службовими витратами й без необхідності тримати стан з’єднання. Протоколи прикладного рівня (CoAP, MQTT-SN) зазвичай працюють поверх UDP, оскільки втрата окремих повідомлень часто не критична й може бути компенсована наступною передачею, а можливість мультикасту дозволяє відразу надсилати оновлення одразу великій групі вузлів без встановлення багатьох TCP-з’єднань. Ще один приклад масового використання UDP — система DNS: для коротких запитів достатньо одного дейтаграми без затримок на handshake, а у випадку втрати DNS-клієнт просто повторює запит або, якщо відповідь надто велика (наприклад, зона-трансфер), переходить на TCP. Аналогічно, протоколи mDNS чи SSDP використовують можливість широкомовлення UDP-пакетів у локальній мережі: навіть якщо частина повідомлень загубиться, періодичне повторювання забезпечує необхідний рівень надійності, водночас знижуючи загальне мережеве навантаження. Таким чином, UDP, незважаючи на «незахищеність» каналів і відсутність гарантій доставки, залишається базовим інструментом для тих випадків, коли критичною є швидкість і мінімальні ресурси, а невелика втрата даних може бути перетворена на прийнятну або взагалі непомічену кінцевим додатком.

2.1.4 Сучасні розширення та альтернативи (QUIC, MPTCP, DCCP, UDPLite)

На сьогодні TCP і UDP продовжують розвиватися, а поряд з ними з’явилися нові протоколи та розширення, що поєднують їхні переваги або задовольняють нішеві потреби — наприклад, низькі затримки, мультиплексування, багатошляховість чи гнучка обробка помилок.

QUIC (Quick UDP Internet Connections) працює поверх UDP і стандартизований у RFC 9000. Він інтегрує шифрування TLS 1.3 безпосередньо в транспортний рівень, тому встановлення безпечного з’єднання вимагає лише 0–1 RTT проти кількох RTT у випадку HTTPS/TCP. Це дозволяє починати обмін даними майже миттєво (а при повторних підключеннях — з 0-RTT). QUIC підтримує мультиплексування кількох потоків всередині одного з’єднання: якщо один потік втрачає пакет, інші продовжують передавання без блокування (уникаючи head-of-line блокування, характерного для HTTP/2 over TCP). Для контролю заторів QUIC успадковує алгоритми TCP (зазвичай CUBIC, але можливу підтримку BBR та інших), а також дозволяє міграцію з’єднання (перехід між IP/портами без розриву), що корисно на мобільних пристроях при перемиканні між Wi-Fi і стільниковою мережею. Уже сьогодні QUIC лежить в основі HTTP/3 і активно використовується браузерами та CDN, прискорюючи завантаження вебсторінок і підвищуючи стійкість до втрат.

Multipath TCP (MPTCP) — розширення TCP (RFC 8684), яке дозволяє одному логічному з’єднанню одночасно використовувати кілька мережевих шляхів. Після узгодження мультишляховості під час рукопотискання відкриваються додаткові subflows — окремі TCP-потоки по різних маршрутах. З погляду програми це прозоро: вона працює з одним сокетом, а MPTCP розподіляє сегменти між доступними шляхами. Завдяки цьому загальна пропускна здатність зростає (якщо передавати дані паралельно по різних

інтерфейсах), а при втраті одного з’єднання (наприклад, через втрату Wi-Fi) трафік миттєво переключається на інший шлях (мобільний інтернет) без розриву сеансу. Це особливо корисно для смартфонів, які одночасно використовують Wi-Fi і стільникову мережу, а також у серверних рішеннях для агрегації каналів. Реалізація MPTCP є в ядрах Linux, iOS та деяких мережевих пристроях, але вимагає підтримки з двох боків з’єднання і може блокуватися старими маршрутизаторами чи брандмауерами.

Datagram Congestion Control Protocol (DCCP) — транспортний протокол (RFC 4340) для застосувань, що потребують контролю заторів, але не гарантії доставки чи впорядкування. Він встановлює безпідтверджуване з’єднання: є handshake і нумерація датаграм, але пакети не підтверджуються повторно. Головна перевага — можливість вибору алгоритму контролю заторів (CCID profile), наприклад CCID2 (аналог TCP Reno) або CCID3 (TCP-Friendly Rate Control). Це дозволяє мультимедіа-додаткам адаптувати передачу так, щоб уникнути надмірного завантаження мережі (щоб стрім не «вимикав» інтернет для інших), але не затримуватися на ретрансляціях: якщо кадр відео не дійшов, додаток може просто пропустити його, а DCCP автоматично знижує швидкість до стабільного рівня. DCCP залишається нішевим і не набув широкого поширення, але важливий як спроба адаптувати транспортний рівень до потреб потокових медіа.

UDPLite (RFC 3828) — варіація UDP з гнучкішою перевіркою контрольної суми. Замість поля «довжина» він має поле Coverage, що визначає, яка частина датаграми захищена контрольною сумою. Бітові помилки за межами цієї критичної області ігноруються, тобто якщо другорядні дані пошкоджено, пакет усе одно доставляється, а додаток може вирішити, як їх обробити. Це корисно для аудіо/відео з похибками або сенсорних даних, де краще отримати «неідеальні» дані, ніж не отримати нічого. UDPLite зменшує кількість повторних передач, економить пропускну здатність і енергію, що важливо у бездротових мережах і мультимедіа.

Отже, кожне з цих рішень відповідає на конкретні обмеження TCP чи UDP: QUIC мінімізує затримки й одразу шифрує трафік, MPTCP підвищує продуктивність і стійкість завдяки багатошляховості, DCCP вводить адаптивне керування заторами для потокових даних без гарантій доставки, а UDPLite дозволяє передавати частково пошкоджені дані. Всі вони працюють поверх IP і адресують ті випадки, де класичний TCP чи UDP в чистому вигляді не є оптимальними. Вибір залежить від вимог: час встановлення з'єднання, чутливість до втрат, рівень необхідної надійності, безпека та можливості мережі.

2.1.5 Порівняльний аналіз і рекомендації щодо вибору протоколу

У різних мережевих умовах оптимальним може бути різний транспортний протокол. TCP та UDP мають перевірені переваги й недоліки, а новіші (QUIC, MPTCP, DCCP, UDPLite) поєднують їхні плюси або пропонують нові можливості. Нижче узагальнено ключові характеристики протоколів і рекомендації щодо їх вибору.

Затримка встановлення з'єднання. TCP вимагає трикрокового рукостискання, а з HTTPS – ще й TLS, що додає 2–3 RTT. UDP не встановлює з'єднання – передача починається миттєво. QUIC поєднує транспортне й TLS-рукостискання, скорочуючи стартову затримку до 0–1 RTT. Це особливо вигідно для HTTP/3. MPTCP додає незначні накладні витрати до TCP, DCCP вимагає ACK, а UDPLite – як UDP. Отже, найменші затримки старту забезпечують UDP і QUIC.

Надійність і порядок. TCP гарантує доставку й порядок байтів, QUIC робить те саме поверх UDP. MPTCP також забезпечує повну надійність. UDP, DCCP і UDPLite ненадійні – не гарантують доставку чи порядок, хоч DCCP виявляє втрати. Для гарантованої доставки обирають TCP, MPTCP або QUIC; для додатків, що витримують втрати, – UDP чи UDPLite.

Контроль заторів. TCP має вбудований контроль (AIMD, Slow Start), MPTCP адаптується до кількох маршрутів. QUIC використовує ті самі алгоритми, дотримуючись принципу TCP-friendly. DCCP пропонує профілі для різних типів трафіку. UDP і UDPLite – без вбудованого контролю, тому додатки повинні регулювати бітрейт самостійно (як у RTP/RTCP). Найбезпечніші для мережі – TCP, MPTCP, QUIC, DCCP.

Безпека і NAT. TCP і UDP не шифрують трафік самостійно (TLS/DTLS потрібен окремо). QUIC забезпечує вбудоване шифрування (TLS 1.3). MPTCP передає як звичайний, так і TLS-трафік. DCCP теоретично підтримує DTLS, але це рідко реалізується. QUIC і UDP легко проходять через NAT завдяки стандартним портам, TCP – традиційно сумісний. MPTCP іноді блокується через нові TCP-опції, DCCP і UDPLite можуть бути невідомими для мережевого обладнання.

Адаптивність. TCP – лише point-to-point і нестійкий до зміни IP. MPTCP підтримує кілька маршрутів одночасно. UDP дозволяє змінювати адресу, але потребує ручної синхронізації. QUIC має вбудовану підтримку мобільності. QUIC також підтримує мультиплексування потоків в одному з’єднанні, UDP – за потреби реалізується на рівні додатку. UDP та UDPLite підтримують multicast, інші – ні.

Рекомендації:

- Веб-сервіси: QUIC/HTTP/3 – швидкий старт, шифрування, мультиплексування.
- Мобільні пристрої: MPTCP – одночасне використання кількох інтерфейсів без втрати сесії.
- VoIP і відео: UDP + RTP/RTCP – мінімальна затримка, при потребі – QUIC.
- Стрімінг: UDP з SRT або QUIC – контроль, надійність, шифрування.
- IoT: UDP з CoAP або MQTT-SN – легковагові протоколи з низькими накладними витратами.

- DNS: UDP – мінімальна затримка, TCP – для великих відповідей.

Таким чином, вибір протоколу залежить від вимог: гарантії доставки – TCP, QUIC; мінімальна затримка – UDP; мобільність і багатоканальність – MPTCP; потокове медіа – SRT/QUIC; multicast – UDP. Інфраструктурна підтримка також важлива – не всі мережі підтримують нові протоколи. Інженер має врахувати затримку, надійність, безпеку й сумісність при виборі транспорту.

2.2. Протоколи, орієнтовані на мультимедіа: RTP, RTMP, RTSP

2.2.1 Протокол RTP (Real-Time Transport Protocol)

Протокол RTP (англ. Real-Time Transport Protocol) є основним стандартом для передавання мультимедійної інформації (аудіо та відео) в режимі реального часу через IP-мережі. Розроблений робочою групою Audio/Video Transport в межах IETF, він вперше був стандартизований у RFC 1889 (1996) і пізніше оновлений до RFC 3550 (2003). На відміну від TCP, RTP працює переважно поверх UDP, що дозволяє мінімізувати затримки, навіть якщо при цьому можливі втрата або перестановка пакетів.

Завдяки підтримці multicast-трансляцій RTP широко використовується в системах групового мовлення, наприклад, для онлайн-конференцій або корпоративного IPTV. Водночас, для забезпечення зворотного зв’язку щодо якості з’єднання й синхронізації потоків, він зазвичай використовується у парі з протоколом RTCP (Real-Time Control Protocol), що забезпечує збирання статистики про втрати, затримки та jitter.

Важливо зазначити, що RTP не виконує функції встановлення або завершення сесії – це покладається на сигнальні протоколи вищого рівня, такі як SIP, H.323 чи RTSP, а також на SDP (Session Description Protocol), який описує параметри медіасесії.

Структура заголовка RTP

Заголовок RTP має фіксовану довжину 12 байтів (96 біт) і побудований із 32-бітових слів. Він містить низку полів, необхідних для ідентифікації, впорядкування та синхронізації медіапотоку. Нижче наведено опис основних компонентів:

- Version (V, 2 біти): поточна версія протоколу RTP (наразі V=2).
- Padding (P, 1 біт): вказує на наявність додаткових байтів заповнення в кінці пакета.
- Extension (X, 1 біт): сигналізує про присутність розширеного заголовка.
- CSRC Count (CC, 4 біти): кількість ідентифікаторів джерел, що беруть участь у складанні потоку.
- Marker (M, 1 біт): інтерпретується на рівні застосування (наприклад, межа кадру відео).
- Payload Type (PT, 7 біт): тип корисного навантаження (вказує на кодек чи формат даних).
- Sequence Number (16 біт): інкрементується з кожним пакетом і дає змогу виявляти втрати та відновлювати порядок.
- Timestamp (32 біти): фіксує момент часу захоплення медіаданих.
- SSRC (32 біти): унікальний ідентифікатор джерела потоку.
- CSRC List (0–15 полів по 32 біти): список джерел, що зробили внесок у потік.
- Header Extension (опційно): містить додаткові дані для застосунку.

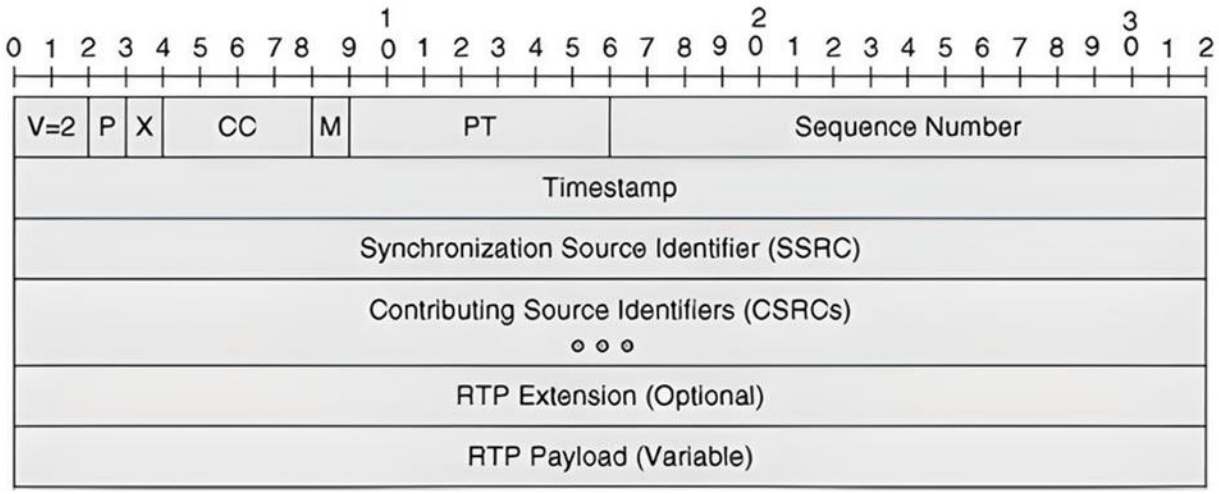

Рисунок 2.1 – Формат заголовка RTP (32-бітові слова). Поля включають версію (V=2), біти P, X, M, лічильник CSRC, тип корисних даних (PT), порядковий номер, мітку часу, ідентифікатори джерел синхронізації (SSRC) та перелік додаткових джерел (CSRC)

Безпека RTP

Оскільки базова специфікація RTP не включає механізмів шифрування чи автентифікації, для захисту даних використовується розширення SRTP (Secure RTP), визначене в RFC 3711. SRTP забезпечує шифрування, контроль цілісності (HMAC) та захист від повторного передавання пакетів. У сучасних системах на основі WebRTC SRTP використовується за замовчуванням разом із DTLS для обміну ключами. Без такого захисту RTP-потоки є вразливими до прослуховування та підміни, оскільки поле SSRC не перевіряється й може бути сфальсифіковане.

Сфери застосування

RTP широко використовується в системах VoIP (зокрема, у протоколах H.323, SIP), відеоконференціях, IP-телебаченні та потоковому передаванні мультимедіа. Його реалізовано в багатьох програмних бібліотеках і фреймворках — GStreamer, FFmpeg, а також у WebRTC API. Проте у браузерах RTP не доступний напряму – він використовується лише як частина WebRTC, а обробка відбувається на нижчому рівні, прозоро для розробника.

На відміну від вебтехнологій, RTP підтримується напряму в медіаплеєрах (наприклад, VLC, FFplay), однак на мобільних пристроях його підтримка зазвичай вимагає встановлення додаткових застосунків.

Масштабованість і ефективність

RTP має вбудовану підтримку multicast, що дозволяє значно зменшити навантаження на сервер при трансляції одному або багатьом користувачам. Проте глобальне використання multicast у відкритому Інтернеті обмежене через складність маршрутизації та низьку підтримку з боку обладнання. Тому найчастіше RTP застосовується в унікаст-режимі, де кожен клієнт отримує окремий потік. Це підходить для невеликих конференцій і дзвінків, але менш ефективне при великій кількості глядачів.

Для масштабного мовлення (тисячі користувачів) сучасні системи надають перевагу технологіям потокової доставки через HTTP, які краще інтегруються з CDN та кешуванням. Однак RTP зберігає свою актуальність у випадках, де критична низька затримка і потрібна точна синхронізація мультимедійних потоків.

Висновок

RTP залишається незамінним стандартом для передачі мультимедіа в режимі реального часу. Його переваги включають підтримку синхронізації, multicast, розширюваність і широке використання у VoIP і WebRTC. Водночас для забезпечення безпеки і масштабованості в багатокористувацьких середовищах протокол потребує доповнення — зокрема, SRTP та відповідної інфраструктури ретрансляції або альтернативних протоколів потокового мовлення.

2.2.2 Протокол RTMP (Real-Time Messaging Protocol)

Протокол RTMP (англ. Real-Time Messaging Protocol) був створений компанією Macromedia (пізніше придбаною Adobe) як транспорт для потокової передачі аудіо, відео та службових даних між Flash-програвачем і

медіасервером. У 2000-х роках RTMP став технологічною основою для Flash Video і активно використовувався у вебстрімінгу. Незважаючи на те, що підтримку Flash припинено у 2020 році, RTMP залишається актуальним на етапі першої милі (від енкодера до платформи), зокрема на сервісах типу YouTube та Facebook Live.

RTMP працює поверх TCP (типово порт 1935) і підтримує постійне дуплексне з'єднання з низькою затримкою. Завдяки механізмам фрагментації і мультиплексування він забезпечує синхронну передачу кількох типів даних через єдиний TCP-потік. У 2009 році Adobe частково відкрила специфікацію протоколу, описавши базову логіку його функціонування.

Варіанти RTMP:

- RTMP (plain): базовий, незашифрований варіант на порту 1935.
- RTMPS: RTMP поверх TLS, використовує порт 443 або 1935 (з TLS), забезпечує повноцінне шифрування трафіку.
- RTMPE: пропрієтарний метод шифрування від Adobe на базі алгоритму Диффі-Геллмана; менш захищений, ніж RTMPS.
- RTMPT: тунелювання RTMP через HTTP/HTTPS, дає змогу обходити файрволи, використовуючи порти 80 і 443.
- RTMFP: UDP-альтернатива RTMP з підтримкою P2P-взаємодії; не отримала широкого поширення.

Структура RTMP-протоколу та формат заголовків

Перед початком передачі клієнт і сервер обмінюються повідомленнями рукостискання: C0–C2 і S0–S2. Після успішного встановлення TCP-з'єднання RTMP передає дані у вигляді повідомлень (messages), які фрагментуються на чанки (chunks). Чанки різних потоків можуть перемежовуватись в одному з'єднанні, що забезпечує ефективне мультиплексування аудіо, відео та службових даних.

Кожен чанк містить заголовок, що складається з таких компонентів:

- Basic Header (1–3 байти): вказує формат заголовка (fmt) та ідентифікатор каналу (Chunk Stream ID).

- Message Header (0, 3, 7 або 11 байтів): несе метадані про повідомлення (мітка часу, довжина, тип, ID потоку).
- Extended Timestamp (0 або 4 байти): додається при перевищенні 3-байтової межі часу (≥ 16777215).
- Chunk Data: частина корисного навантаження, розбита згідно з узгодженим розміром чанка (типово 128 байт).

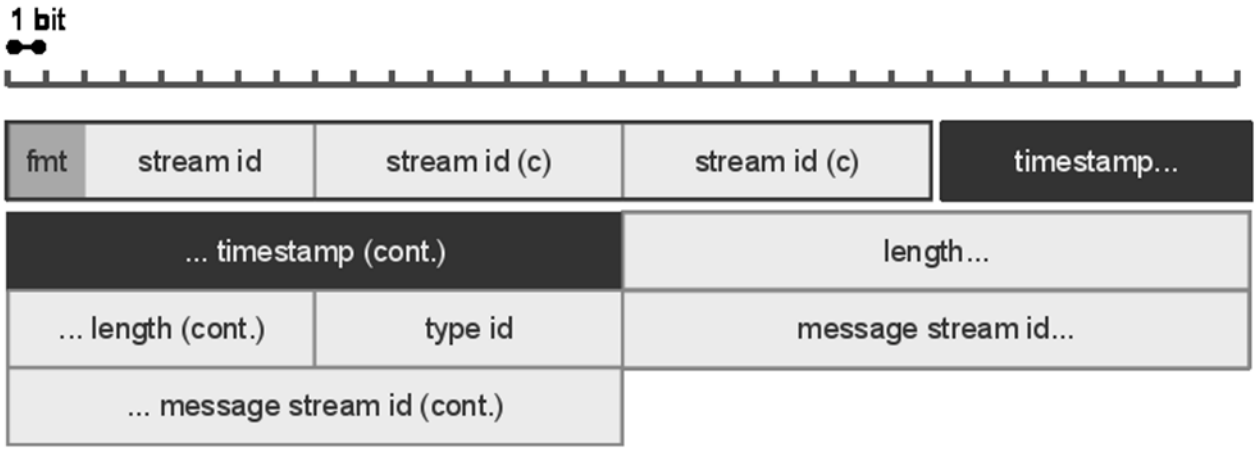

Рисунок 2.2 – Схема заголовків RTMP-чанка (Chunk Header). Заголовок складається з Basic Header (fmt і Chunk Stream ID), Message Header (Timestamp, Length, Type ID, Message Stream ID) та, за потреби, Extended Timestamp. Після заголовка передається Chunk Data — фрагмент корисного повідомлення

Управління передачею та службові повідомлення

RTMP підтримує службові повідомлення (Protocol Control Messages), які регулюють потік і з’єднання:

- Ping: вимірювання затримки (RTT).
- Acknowledgement (Ack): підтвердження прийому певного обсягу даних.
- Set Peer Bandwidth, Window Acknowledgement Size: контроль пропускної здатності та вікна з підтвердженнями.

Додатково підтримуються команди керування на базі AMF (Action Message Format) — наприклад, відтворення, пауза, автентифікація, передача метаданих.

Безпека RTMP

Базовий RTMP не забезпечує шифрування. RTMPS, як TLS-альтернатива, забезпечує повну конфіденційність трафіку та підтримується основними стрімінговими платформами. RTMPE, хоч і забезпечує шифрування без сертифікатів, має нижчий рівень безпеки. Автентифікація реалізується через параметри сесії (логін, пароль, токен), передані у connect-команді або в URL потоку. Без шифрування такі облікові дані можуть бути перехоплені, тому RTMPS вважається єдиним надійним методом захисту.

Сумісність і підтримка

Історично RTMP пов’язаний із Flash, який дозволяв його використання в браузерах. Після припинення підтримки Flash прямий перегляд RTMP у браузерах став неможливим. Натомість, RTMP сьогодні використовується на рівні між енкодером (наприклад, OBS) і сервером, після чого конвертується у HLS/DASH для дистрибуції. RTMP не підтримується нативно на iOS/Android, однак реалізується через сторонні бібліотеки: ExoPlayer з RTMP-модулем, VLC, FFmpeg тощо.

Масштабованість та роль RTMP у сучасних архітектурах

RTMP-з’єднання є постійним і станутворюючим, що обмежує його масштабування порівняно з HTTP. Щоб обслуговувати велику кількість клієнтів, впроваджуються каскадні медіасервери (Wowza, Red5, Adobe Media Server), а також CDN-вузли, що приймають RTMP і транслюють ближче до глядача. Проте для глобального масштабування сучасні системи переходять на HTTP-протоколи.

Типова сучасна архітектура виглядає так: RTMP-потік подається енкодером на Ingress-сервер, далі перекодується у HLS або DASH, які передаються кінцевому користувачу через CDN. Таким чином, RTMP використовується переважно як протокол першої милі. Він залишається

актуальним для професійного стрімінгу завдяки надійності, простоті реалізації та низькій затримці, попри поступове витіснення сучаснішими протоколами для доставки контенту на останній милі.

2.2.3 Протокол RTSP (Real-Time Streaming Protocol)

Протокол RTSP (англ. Real-Time Streaming Protocol) є прикладним мережевим протоколом, призначеним для керування потоковими мультимедійними сесіями між клієнтом і сервером. Його функціональність нагадує пульт дистанційного керування: він дозволяє запускати, призупиняти або зупиняти відтворення аудіо- та відеопотоків. Сам протокол не займається передачею медіаданих напряму – для цього він зазвичай використовується у зв’язці з RTP/RTCP. Стандартизований у RFC 2326 (версія 1.0, 1998 р.) та оновлений до RFC 7826 (версія 2.0, 2016 р.). RTSP активно використовується у відеоспостереженні, VoD-системах, а також у локальних мережах для медіаконтролю.

Принципи роботи та логіка протоколу RTSP

RTSP — це станутворюючий протокол: після встановлення сесії з ідентифікатором Session ID сервер зберігає її контекст. На відміну від HTTP, RTSP дозволяє не лише запит-відповідь, але й асинхронні повідомлення. Типова послідовність взаємодії виглядає наступним чином:

1. OPTIONS — клієнт дізнається, які методи підтримує сервер.
2. DESCRIBE — запит на отримання SDP-опису медіапотоку (кодеки, треки, порти, формати).
3. SETUP — налаштування транспортного каналу для медіа (TCP/UDP, multicast/unicast).
4. PLAY — запуск стріму. Можна вказати діапазон відтворення та швидкість (Range, Scale).
5. PAUSE — тимчасове зупинення потоку без розриву сесії.
6. TEARDOWN — завершення сеансу та звільнення ресурсів.

Додаткові команди:

- ANNOUNCE, RECORD, REDIRECT, GET/SET_PARAMETER — розширюють функціональність керування, не всі підтримуються сервером обов'язково.

Формат повідомлень RTSP подібний до HTTP: рядок запиту (метод, URI, версія), заголовки та тіло (у разі DESCRIBE — SDP). Відповіді сервера також текстові з кодами стану (200 OK, 404 Not Found тощо). Кожен запит нумерується через заголовок CSeq, для ідентифікації сесії використовується заголовок Session.

Сумісність і підтримка

На відміну від HTTP, RTSP не підтримується сучасними браузерами напряму. Після відмови від Flash і NPAPI-плагінів пряма інтеграція RTSP в браузер стала неможливою. Для відтворення використовують сторонні застосунки (VLC, IP-камерні клієнти) або шлюзи, які конвертують RTSP у HLS або WebRTC. На мобільних пристроях RTSP також не підтримується на рівні ОС: для відтворення потрібні окремі застосунки.

Незважаючи на це, RTSP залишається стандартом де-факто у сфері відеоспостереження. Більшість IP-камер підтримують видачу відео через RTSP, а стандарт ONVIF прямо вимагає такої підтримки. У сфері VoD-серверів RTSP може використовуватись у локальних мережах із прямим керуванням потоком.

Безпека RTSP

RTSP 1.0 успадковує механізми автентифікації від HTTP: Basic (Base64) та Digest. У разі відсутності автентифікації камери повертають 401 Unauthorized, після чого клієнт має повторити запит із заголовком Authorization. Водночас, базова специфікація не передбачає шифрування: логіни й паролі передаються майже відкритим текстом, що несе загрозу.

Для шифрування використовується RTSPS (RTSP через TLS) — встановлюється захищене з’єднання на порту 322 або 443, через яке передаються як команди, так і потоки (у разі TCP-інтерливінгу). Також

підтримується тунелювання RTSP через HTTPS або використання SRTP для зашифрованого медіатрафіку, залишаючи RTSP-команди окремими.

Масштабованість і сучасне використання

RTSP не передбачає нативну підтримку CDN чи кешування. У великих системах застосовують ієрархічні сервери або multicast, якщо дозволяє мережа. Multicast дозволяє одному серверу транслювати потік усім клієнтам без дублювання, але практично реалізується лише у корпоративних мережах. В Інтернеті multicast не маршрутизується провайдерами, тому масштабування RTSP відбувається через unicast та каскадні сервери.

У сучасній архітектурі RTSP здебільшого використовується на рівні ingest — IP-камери надсилають потоки до хмарних платформ, які далі перекодовують відео в HLS, DASH чи WebRTC для масової доставки. Завдяки низькій затримці та можливості повного контролю над сесією RTSP залишається актуальним у спеціалізованих системах, попри витіснення з фронтенду вебу.

Висновок

RTSP — ключовий протокол для інтерактивного керування потоковим відтворенням у реальному часі. Хоча обмежений у масштабованості та браузерній підтримці, він зберігає важливу роль у відеоспостереженні, професійному відео та локальних медіасерверах. Сучасні розширення (TLS, SRTP) дозволяють використовувати RTSP у безпечних середовищах, а шлюзи забезпечують його інтеграцію в нові веборієнтовані рішення.

2.3 Адаптивні та сучасні протоколи: HLS, DASH, SRT, WebRTC

У сучасному потоковому відеомовленні широко застосовуються протоколи, що забезпечують адаптацію до швидкості мережі та різні рівні затримки. До них належать HTTP Live Streaming (HLS) та MPEG-DASH – HTTP-базовані протоколи адаптивного бітрейту для масового потокового передавання, а також новіші рішення для низької затримки, такі як SRT (Secure

Reliable Transport) та WebRTC (Web Real-Time Communication). Кожен із цих протоколів має свою архітектуру, формати даних та сценарії використання, а також особливості сумісності, адаптивності, безпеки та масштабування. У цьому розділі проаналізовано принципи роботи та характеристики HLS, DASH, SRT і WebRTC, зокрема їхню архітектуру, формати сегментів/маніфестів, практичне застосування, підтримку на різних платформах, здатність до адаптації в умовах зміни мережевих умов, типові затримки, механізми безпеки, а також підтримку CDN і можливості масштабування.

2.3.1 Протокол HLS (HTTP Live Streaming)

HLS (HTTP Live Streaming) — це адаптивний протокол потокового відео, розроблений компанією Apple у 2009 році. Він передає відео у вигляді послідовності коротких медіасегментів через звичайні HTTP-з’єднання.

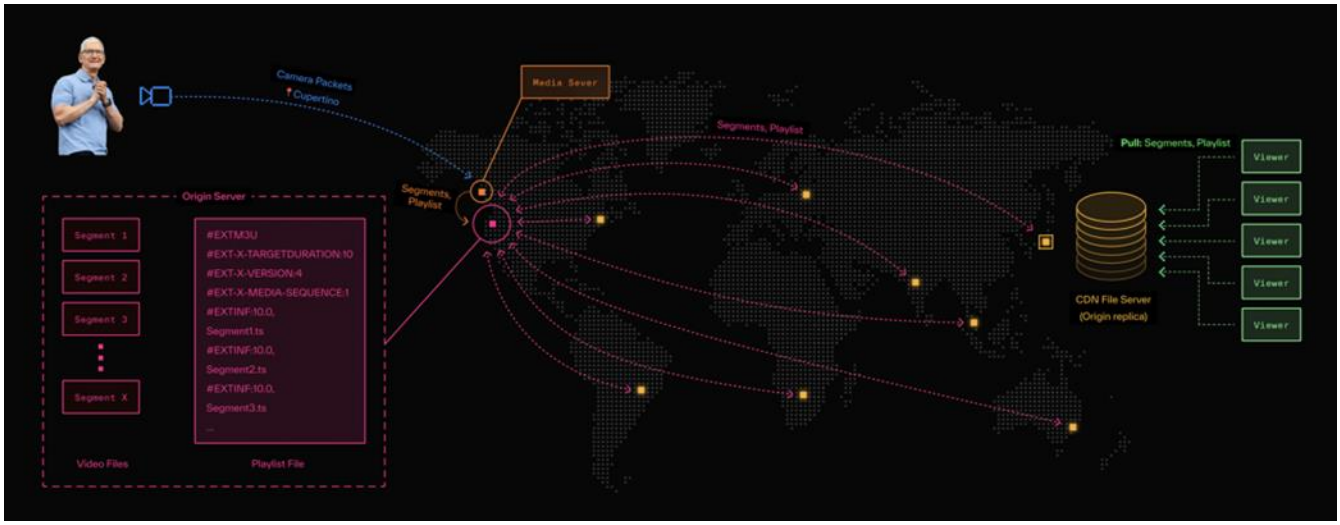

Рисунок 2.3 – Схема HLS: медіасервер розбиває потік на сегменти, формує плейліст і поширює дані через CDN. Клієнт завантажує плейліст і послідовно відтворює сегменти

Джерело сигналу (камера, енкодер) надсилає потік на медіасервер, який розбиває його на сегменти (зазвичай 6 с) у форматі MPEG-TS або fMP4 та

формує плейліст M3U8 із посиланнями. Клієнт регулярно оновлює плейліст і завантажує сегменти для безперервного відтворення. Така архітектура легко масштабується через CDN і добре працює з проксі та фаєрволами.

Формат і сегментування.

HLS використовує текстові плейлісти .m3u8: медіа-плейлісти для одного потоку і майстер-плейлісти для кількох варіантів якості. Сегменти — це короткі фрагменти відео (2–10 с), які завантажуються і відтворюються по черзі. У режимі Low-Latency HLS використовуються часткові сегменти (~200 мс), що знижує затримку.

Адаптивність.

HLS підтримує адаптивний бітрейт: клієнт обирає відповідну якість із майстер-плейліста і може динамічно переключатися між варіантами в залежності від швидкості мережі та буфера. Перемикання здійснюється між сегментами й не помітне користувачу.

Сумісність.

HLS підтримується на iOS/macOS, Android, смарт-ТВ, браузерами (через MSE та hls.js) і є стандартом для відео в App Store. Більшість сучасних платформ (YouTube, Instagram Live, OTT-сервіси) підтримують HLS як основний протокол потокового відео.

Затримка.

Типова затримка залежить від тривалості сегментів. У класичних налаштуваннях — 15–30 с. LL-HLS дозволяє зменшити її до 2–5 с. Затримка компенсується надійністю буферизації та гнучкістю до перебоїв з'єднання.

Безпека та масштабованість.

HLS підтримує HTTPS і шифрування сегментів (AES-128). Ключі передаються через захищені URL, що дозволяє реалізувати просту DRM-систему. Сегменти легко кешуються на CDN, завдяки чому HLS чудово масштабується на мільйони глядачів без перевантаження серверів.

Висновок.

HLS поєднує в собі високу надійність, гнучкість до мережевих умов, адаптивну якість та широку підтримку пристроїв. Завдяки цим властивостям він став де-факто стандартом для відео в Інтернеті — як для трансляцій у прямому ефірі, так і для VoD.

2.3.2 Протокол MPEG-DASH (Dynamic Adaptive Streaming over HTTP)

MPEG-DASH — це відкритий стандарт потокового відео, затверджений MPEG у 2012 році. Як і HLS, DASH працює за принципом сегментації: відео ділиться на короткі фрагменти, які клієнт завантажує по HTTP. Відмінністю є формат маніфесту — XML-документ MPD (Media Presentation Description), що описує структуру медіа, репрезентації (якість), адаптаційні набори (мовні, відео/аудіо доріжки) та сегменти.

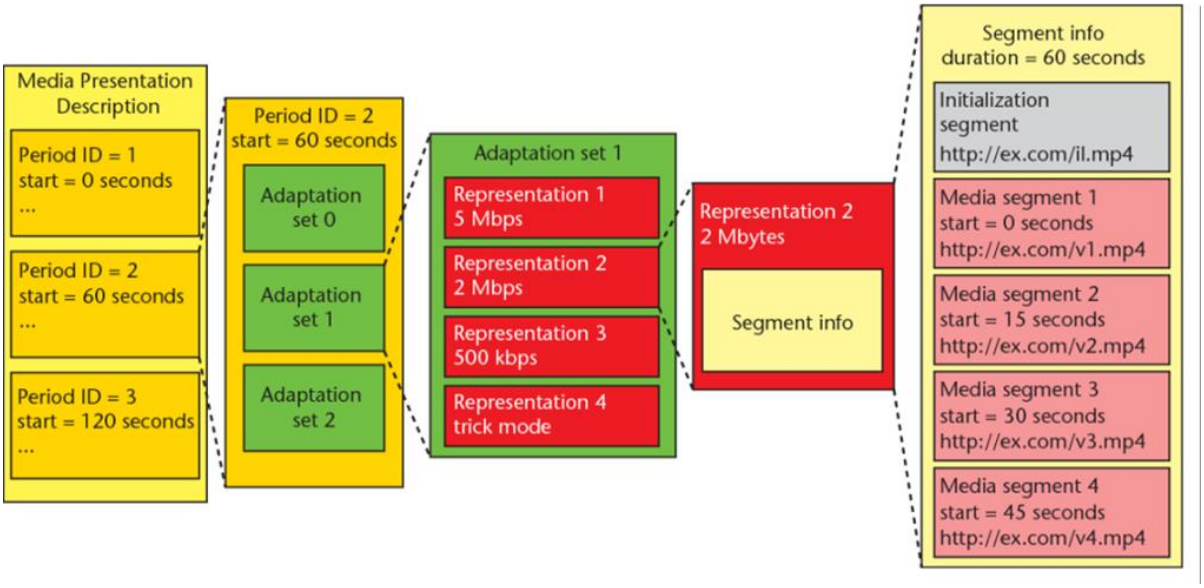

Рисунок 2.4 – Структура MPD: періоди, адаптаційні набори, репрезентації та сегменти відео з відповідними параметрами якості

Сегментування і контейнери.

Сегменти зазвичай мають тривалість 2–6 с і зберігаються у форматі fMP4 (CMAF). Клієнт використовує .mpd для завантаження сегментів з

вибраної репрезентації. DASH підтримує byte-range запити та ініціалізаційні сегменти. Перемикання між якостями відбувається на межах сегментів.

Адаптивність.

Клієнтський плеєр (наприклад, dash.js) оцінює швидкість мережі та буфер, динамічно обираючи найкращу якість. Алгоритм ABR не жорстко заданий стандартом — реалізація може варіюватися.

Сумісність.

DASH працює з HTML5, Media Source Extensions (MSE) та Encrypted Media Extensions (EME). Підтримується в Chrome, Firefox, Edge, Safari (на macOS), Android-пристроях і смарт-ТВ. На iOS потрібен fallback до HLS через обмеження Safari.

Практичне використання.

YouTube і Netflix використовують MPEG-DASH як основний протокол потокового мовлення. DASH підтримується багатьма плеєрами: ExoPlayer, Bitmovin, THEO, а також індустрією через DASH Industry Forum.

Затримка.

Типова затримка складає 10–20 с. LL-DASH на основі CMAF і chunked transfer дозволяє знизити її до 2–3 с, використовуючи часткові сегменти та HTTP/2/3.

DRM та безпека.

DASH підтримує Common Encryption (CENC), сумісний із Widevine, PlayReady, FairPlay. Працює через EME у браузерах. Уніфіковане шифрування спрощує інтеграцію кількох DRM-систем без перекодування.

Масштабованість.

Сегменти DASH кешуються на CDN, забезпечуючи масштабування до мільйонів глядачів. DASH використовується як основний протокол для масштабованих трансляцій із високою якістю і підтримкою сучасної інфраструктури.

Висновок.

MPEG-DASH — універсальний, гнучкий і стандартизований протокол, що підтримує адаптивне потокове відео з мінімальною затримкою, широкою сумісністю та інтеграцією DRM. Це надійна альтернатива HLS для сучасного потокового мовлення.

2.3.3 Протокол SRT (Secure Reliable Transport)

SRT (Secure Reliable Transport) — це відкритий транспортний протокол, розроблений компанією Haivision у 2017 році для надійної доставки відео з низькою затримкою через ненадійні мережі, переважно Інтернет. На відміну від HLS або DASH, SRT не працює з файлами чи сегментами — він забезпечує передачу безперервного відеопотоку «від пункту до пункту» (point-to-point), що особливо важливо для віддалених включень, прямих трансляцій або обміну відеосигналами між серверами. У 2017 році Haivision відкрила вихідний код протоколу, і була створена спільнота SRT Alliance, яка на сьогодні об’єднує сотні компаній, серед яких Microsoft, AWS, Alibaba, ESPN та інші.

Архітектура і принцип роботи. SRT працює поверх транспортного протоколу UDP, додаючи на прикладному рівні надійні механізми, подібні до TCP, але з орієнтацією на низьку затримку. Він використовує нумерацію пакетів, підтвердження доставки (ACK), негативні підтвердження (NAK) та повторну передачу загублених пакетів (ARQ – Automatic Repeat reQuest). Якщо приймач виявляє втрату (прогалину в послідовності пакетів), він надсилає NAK-повідомлення, після чого відправник із буфера повторно передає лише втрачені дані. Важливо, що SRT не зупиняє основний потік, як TCP — повтор передається паралельно.

Ще однією ключовою особливістю є контрольована затримка. Перед початком передачі встановлюється розмір буфера (зазвичай 120–250 мс), який визначає цільову затримку та дозволяє компенсувати короткочасні втрати або коливання мережі. Якщо мережа погіршується, SRT може адаптувати розмір буфера. Якщо якийсь пакет не встигає прибути вчасно (до вичерпання буфера),

він може бути відкинутий, щоб уникнути зростання затримки (conditional too-late dropping).

Передача відео та формат даних. Протокол є «прозорим» до типу вмісту: зазвичай він транспортує стиснені відео- або аудіодані у форматі MPEG-TS або навіть RAW. Ці дані інкапсулюються в власні пакети SRT, які містять службову інформацію (номер пакету, мітку часу, контрольну суму тощо). На відміну від HLS чи DASH, які потребують маніфестів і сегментів, SRT працює з потоком у реальному часі без попередньої підготовки.

Під час встановлення з’єднання SRT узгоджує ролі Caller (ініціатор) і Listener (приймач), налаштовує параметри передачі, включаючи режим корекції помилок (ARQ або ARQ+FEC), розмір буфера, таймаути та шифрування. Передача службових повідомлень (ACK, NAK, KEEPALIVE, HANDSHAKE) реалізована в тих самих UDP-датаграмах, що спрощує роботу з NAT та брандмауерами — достатньо одного порту для всієї сесії.

Продуктивність і адаптивність. SRT забезпечує стабільну передачу навіть у складних умовах мережі. Протокол переносить до 10–20% втрат без помітної деградації відео. Завдяки буферу затримки 0.5–2 секунди досягається плавне відтворення з затримкою менше 1–2 с. SRT динамічно адаптує частоту повторів (NAK), швидкість передачі (при перевантаженні приймача), а також може увімкнути FEC, якщо втрати стають надто високими.

Безпека. SRT має вбудоване шифрування — підтримується AES-128 та AES-256. Ключі можуть бути узгоджені заздалегідь або обмінюватися в процесі з’єднання. Завдяки цим механізмам SRT підходить для захищених трансляцій — новин, спортивних подій, конференцій. Крім того, оскільки протокол використовує лише один UDP-порт, його легко інтегрувати в корпоративні мережі з жорсткими політиками безпеки.

Масштабування і використання в CDN. Протокол не призначений для прямого широкомовного мовлення: кожен глядач потребує окремого SRT-з’єднання. Тому SRT використовується на першій ділянці (first mile): наприклад, передача з камери до продакшн-серверу, або між серверами. Для

масової доставки (many-to-many) потік перекодується у HLS/DASH. Також можливі сценарії ретрансляції через вузли: один сервер приймає потік по SRT і розповсюджує далі (tree-like схема).

Сценарії використання. SRT застосовується багатьма компаніями у професійних трансляціях. Серед прикладів — ESPN, Al Jazeera, Microsoft, OBS Studio, vMix, Larix Broadcaster. Хмарні сервіси, як AWS та Azure, підтримують SRT-входи. Більшість апаратних енкодерів (Haivision, Teradek, NewTek) мають SRT-режим. Хоча SRT не підтримується браузерами напряму, його можна інтегрувати через шлюзи або програми-програвачі (VLC, PlayerCast).

Висновок. SRT — ефективний протокол реального часу, який поєднує гнучкість UDP з надійністю ARQ, адаптивністю до мережевих умов і високим рівнем безпеки. Він не замінює HLS/DASH у клієнтському сегменті, але є галузевим стандартом для передачі відео між джерелом і сервером у професійному стримінгу.

2.3.4 WebRTC (Web Real-Time Communication)

Архітектура та принцип дії WebRTC. WebRTC – це комплекс технологій реального часу, що дозволяє організовувати передачу аудіо, відео та даних безпосередньо між браузерами або іншими пристроями без встановлення додаткових плагінів. Проєкт WebRTC був започаткований компанією Google у 2011 році та пізніше стандартизований організаціями W3C та IETF, ставши відкритим і безкоштовним для впровадження. Основна ідея WebRTC – це прямий обмін даними peer-to-peer між клієнтами (парами пристроїв), що забезпечує мінімальну затримку при передачі потоків аудіо та відео в реальному часі.

Для досягнення такого рівня взаємодії WebRTC використовує цілу низку технологій та протоколів. Ключовими серед них є: ICE (Interactive Connectivity Establishment), який у поєднанні з STUN і TURN дозволяє клієнтам знаходити один одного в мережі, обходити NAT і брандмауери; DTLS (Datagram

Transport Layer Security), що відповідає за шифрування з’єднання і узгодження ключів; SRTP (Secure Real-Time Transport Protocol) для безпечної передачі медіа, а також API RTCPeerConnection, який інкапсулює всі ці механізми в зручний інтерфейс браузера.

Архітектурно кожен клієнт (зазвичай браузер) має вбудований стек протоколів WebRTC. Коли один користувач хоче встановити відеозв’язок з іншим, процес починається із сигналізації — обміну початковими даними про медіапотоки, параметри з’єднання, кодеки, і потенційні мережеві адреси (кандидати), що передаються у вигляді SDP-повідомлень через допоміжний канал (наприклад, WebSocket або HTTP POST). Далі запускається процедура ICE, під час якої клієнти випробовують різні мережеві шляхи, намагаючись встановити пряме UDP-з’єднання. Якщо прямий зв’язок неможливий, використовується TURN-сервер як проміжна точка ретрансляції. Після встановлення маршруту, трафік шифрується через DTLS, а потім передається у вигляді RTP-потоків, захищених SRTP.

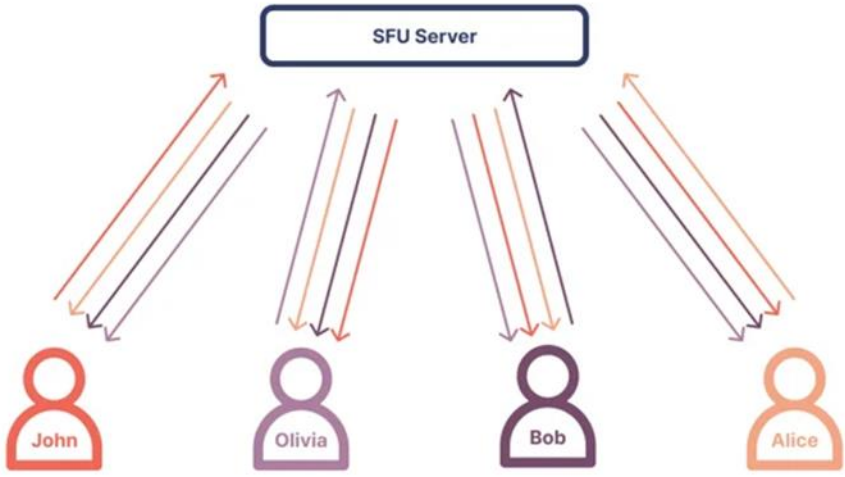

Рисунок 2.5 – Схема топології Selective Forwarding Unit (SFU) для багатоучасникового відеодзвінка WebRTC: кожен клієнт надсилає свій медіапотік лише один раз на SFU-сервер, який потім розсилає копії цього потоку іншим учасникам

Топологія і масштабування. В залежності від кількості учасників та вимог до продуктивності, WebRTC може працювати в одному з кількох режимів: простий mesh (кожен з'єднаний з кожним — P2P), через SFU (Selective Forwarding Unit) — сервер, який отримує потоки і пересилає їх без перекодування, або MCU (Multipoint Control Unit) — сервер, що об'єднує всі потоки в один. Для дзвінків 1-на-1 WebRTC працює напряму, без участі серверів (крім STUN/TURN), що мінімізує затримку і зменшує навантаження на інфраструктуру. Проте у групових відеоконференціях з 3+ учасниками використання SFU дозволяє уникнути квадратичного росту кількості потоків та оптимізувати пропускну здатність клієнтів.

Практичні сценарії використання. WebRTC використовується в багатьох сферах: від простих відеочатів (Google Meet, Facebook Messenger, WhatsApp Web) до складних корпоративних систем відеозв'язку (Zoom, Microsoft Teams). Також він лежить в основі інтерактивних стрімінгів, сервісів підтримки клієнтів, онлайн-освіти та P2P-ігор. Крім медіа, WebRTC дозволяє передавати довільні дані через DataChannel, реалізований поверх SCTP, що відкриває шлях до побудови складних взаємодій без серверів — від спільного редагування до обміну файлами.

Адаптивність та якість. Однією з ключових переваг WebRTC є його здатність до адаптації до умов мережі. Протокол RTP, що використовується для передачі медіа, дозволяє здійснювати моніторинг параметрів мережі в реальному часі (втрати, джиттер, затримка), а вбудовані алгоритми управління конгестією (congestion control) динамічно регулюють бітрейт, частоту кадрів та роздільність відео залежно від поточної ситуації. Це забезпечує стабільну якість зображення навіть у випадках погіршення каналу, уникнення буферизації та підтримку мінімально можливої затримки без втрати з'єднання.

Затримка і відновлення. Завдяки принципу fire-and-forget, WebRTC не повторює втрачені пакети (як TCP), а натомість покладається на алгоритми відновлення на рівні кодеків (PLI, FEC, Insertable Streams). Типова end-to-end затримка при гарному з'єднанні становить 100–300 мс, у складніших умовах

— близько 500 мс, що значно нижче порівняно з HLS або DASH. Внаслідок цього WebRTC активно застосовується у сферах, де критично важлива швидкість реакції: від керування дронами до інтерактивних ігор.

Безпека. WebRTC спочатку проектувався як безпечний протокол. Усі передані медіапотоки шифруються за допомогою SRTP, а ключі обмінюються через DTLS. Важливо, що WebRTC вбудовано в браузер, тому користувач завжди контролює доступ до камери й мікрофона — доступ можливий лише після явно вираженої згоди. Крім того, WebRTC підтримує механізми запобігання витоку IP-адрес (модифікації ICE), та механізми анонімізації трафіку (наприклад, використання TURN-серверів у режимі relay-only).

Масштабування і CDN. На відміну від HLS/DASH, WebRTC не використовує кешування і не працює через CDN. Для масштабування до великої кількості клієнтів (сотні/тисячі) застосовуються SFU або гібридні рішення: WebRTC для ключових учасників, HLS — для пасивної аудиторії. Деякі сервіси використовують хмарні SFU-сервери (Jitsi, Mediasoup, Janus), що дозволяє масштабувати конференції без значної затримки, але з додатковими інфраструктурними витратами.

Висновок. WebRTC — потужний і гнучкий інструмент для організації зв’язку в реальному часі. Його переваги — низька затримка, наскрізне шифрування, адаптація до мережі, підтримка всіх сучасних браузерів і мобільних платформ — зробили його основою більшості сучасних систем відеозв’язку. Хоча він не призначений для масового стримінгу, але в поєднанні з іншими технологіями WebRTC дозволяє будувати масштабовані, безпечні й інтерактивні мультимедійні платформи нового покоління.

2.3.5 Порівняння протоколів передачі мультимедіа

Загальні принципи порівняння.

Розглянуті протоколи передачі мультимедійного контенту — HLS, MPEG-DASH, SRT та WebRTC — відображають різні підходи до організації

стримінгу залежно від цілей, обмежень та умов середовища. Їх можна порівнювати за низкою ключових характеристик, серед яких особливе значення мають: затримка передачі, адаптивність до зміни якості каналу, здатність масштабуватись на велику аудиторію, рівень підтримки платформ, безпека передачі даних, а також сумісність із CDN-інфраструктурою. Варто підкреслити, що жоден з протоколів не є універсальним рішенням: їх ефективність проявляється лише в контексті певних сценаріїв використання — масові трансляції, інтерактивні сесії, низьколатентне мовлення, професійна відеовиробнича інфраструктура тощо.

Порівняльна характеристика затримки, адаптивності та типу передачі. Найважливішим параметром у випадку живого стримінгу є затримка (latency). Протоколи HLS і MPEG-DASH працюють за принципом буферизації сегментів на клієнтському боці, що забезпечує стійкість до втрат, але тягне за собою типову затримку в діапазоні 15–30 секунд (можливо зниження до 2–5 с у Low-Latency реалізаціях). Натомість SRT і WebRTC орієнтовані на мінімальні затримки: WebRTC забезпечує передачу з latency <500 мс, тоді як SRT — в межах 2–5 с, залежно від конфігурації та якості мережі.

У таблиці 2.3.1 узагальнено ключові аспекти, пов’язані з характером передачі, адаптивністю та типовою затримкою для кожного з протоколів:

Таблиця 2.1 — Порівняння затримки та адаптивності протоколів

Протокол	Характер передачі	Механізм адаптації до мережі	Типова затримка (live)
HLS	HTTP-сегментований потік з буферизацією	Клієнтське перемикання якості (ABR) на межах сегментів	~5–30 с (Low-Latency HLS: ~2–5 с)

Продовження таблиці 2.1

MPEG-DASH	HTTP-сегментований потік з буферизацією	Клієнтське перемикання якості (ABR), CENC для DRM	~5–30 с (LL-DASH: ~2–5 с)
SRT	UDP-потік з ARQ і контрольованою буферизацією	Адаптивне регулювання буфера, компенсація втрат, корекція помилок	~2–5 с (можливо <1 с у tuned)
WebRTC	Peer-to-peer через UDP або SFU	Вбудовані congestion control алгоритми, адаптація енкодера	<0.5 с (практично — ~0.2–0.3 с)

Сумісність, масштабованість та безпека. З позиції охоплення аудиторії, HLS і DASH мають незаперечні переваги завдяки простоті доставки через HTTP/HTTPS, повній сумісності з CDN та можливості кешування сегментів на серверних вузлах. Це робить їх ідеальними для широкомасштабних OTT-рішень та трансляцій, де пріоритет — охоплення та якість. SRT і WebRTC мають інший вектор: вони не сумісні з CDN у класичному розумінні, натомість підтримують передачу у реальному часі. WebRTC є незамінним у браузерних відеодзвінках, а SRT — у професійній студійній інфраструктурі для передачі відео з мінімальною затримкою.

Таблиця 2.2 — Порівняння сумісності, безпеки та масштабованості

Протокол	Підтримка платформ	Безпека	Масштабованість	CDN інтеграція
HLS	Всі платформи, рідна підтримка iOS	AES-128, DRM (FairPlay)	Висока	Повна (через HTTP/CDN)

Продовження таблиці 2.2

MPEG-DASH	Більшість браузерів, крім iOS Safari	DRM (Widevine, PlayReady, FairPlay)	Висока	Повна (через HTTP/CDN)
SRT	Програми (OBS, VLC), обладнання	AES-128/256, end-to-end encryption	Обмежена (ретранслятори)	Відсутня (UDP)
WebRTC	Усі сучасні браузери та мобільні платформи	DTLS + SRTP, обов’язкове шифрування	Обмежена (через SFU)	Відсутня (P2P, SFU)

Аналітичнеузагальнення.

Протоколи HLS та MPEG-DASH демонструють високу ефективність у середовищах, де головною метою є доставлення контенту на широку аудиторію з максимальною надійністю, що реалізується за рахунок кешування, буферизації і попередньої обробки. Їхній недолік — помітна затримка, що робить їх не надто придатними для інтерактивних сценаріїв. SRT і WebRTC, у свою чергу, не призначені для масштабованих однонаправлених трансляцій. Вони ідеальні для реального часу — ситуацій, де критично важливі затримка та синхронізація. При цьому WebRTC дає змогу працювати прямо з браузера без додаткового програмного забезпечення, а SRT забезпечує надійну якість навіть при нестабільному з’єднанні, з можливістю контролю над кожним етапом передачі.

Застосування.

- HLS/MPEG-DASH — ідеальні для телевізійних мовників, OTT-сервісів, платформ VOD та новинних ресурсів, де затримка менш критична, а якість і стабільність — у пріоритеті.
- SRT — підходить для професійних трансляцій, спортивних подій, мобільних репортажів, де важлива якість і контроль при низьких затримках.
- WebRTC — основа для вебконференцій, інтерактивних ігор, телемедицини та інших P2P-сценаріїв.

Усі ці технології мають свою нішу і все частіше комбінуються в складних мультимедійних рішеннях — наприклад, одночасне використання WebRTC для інтерактиву і HLS для широкої трансляції з CDN, що дозволяє досягти гнучкості, надійності та масштабованості одночасно.

2.4 Порівняльна таблиця характеристик протоколів

Таблиця 2.3 — Затримка (типова)

Протокол	Типова затримка
TCP	помірна – встановлення з’єднання 2–3 RTT (десятки–сотні мс)
UDP	дуже низька – немає handshakes, затримка мінімальна
RTP	дуже низька – використовує UDP, додаткові заголовки без суттєвих затримок
RTSP	низька – управляє RTP-потоком, тому латентність подібна до RTP
RTMP	низька–середня – зазвичай ~3–30 с (залежить від буферів та умов)
HLS	висока – ~5–30 с (Low-Latency HLS ~2–5 с)
DASH	висока – ~5–30 с (LL-DASH ~2–5 с)
SRT	низька – ≈2–5 с (настроювана; мінімум ~0.1–0.2 с)
WebRTC	дуже низька – <0.5 с (зазвичай ~0.2–0.3 с)
QUIC	дуже низька – 0–1 RTT (власний TLS 1.3, десятки мс)
MPTCP	помірна – аналогічна TCP (2–3 RTT для встановлення)
DCCP	низька – схожа на UDP (підтримує встановлення з’єднання, але без ретрансляцій)
UDP-Lite	дуже низька – як UDP, без повного захисту даних (допускає пошкодження)

Таблиця 2.4 — Толерантність до втрат

Протокол	Допустимість втрат
TCP	низька – не допускає втрат, гарантує надійну доставку

Продовження таблиці 2.4

UDP	висока – не гарантує доставку, втрати приймаються (переважно для мультимедіа)
RTP	висока – не гарантує доставку, призначений для мультимедіа (не критичне втрата пакетів)
RTSP	висока – мультимедіа-потоки базуються на UDP/RTP, отже допускають втрати
RTMP	низька – працює поверх TCP, гарантує доставку без втрат
HLS	низька – HTTP/TCP, сегмент не відтворюється при втраті, відбувається повторна завантаження
DASH	низька – HTTP/TCP (як HLS)
SRT	висока – використовує ARQ (повторні запити втрачених пакетів), мінімізуючи вплив втрат
WebRTC	висока – UDP з механізмами FEC/NACK, деякі втрати компенсуються (аудіо/відео)
QUIC	низька – надійний (як TCP), втрати автоматично повторюються
MPTCP	низька – надійний (як TCP)
DCCP	висока – не гарантує доставку, орієнтований на мінімальні втрати при потоковій передачі
UDP-Lite	висока – допускає пошкодження в частині даних, не відкидає пакети повністю

Таблиця 2.5 — Адаптивність бітрейту

Протокол	Адаптивність бітрейту
TCP	Ні (потік фіксований, без вбудованої адаптації)
UDP	Ні (немає механізму зміни бітрейту на рівні протоколу)
RTP	Ні (постійна передача пакетів без ABR)
RTSP	Ні
RTMP	Ні (потік фіксований, призначений для стабільного стріму)
HLS	Так – клієнт вибирає якість сегментів (ABR)
DASH	Так – клієнтське переключення якості (ABR)

Продовження таблиці 2.5

SRT	Так – вбудоване регулювання буферу й швидкості (налаштування латентності)
WebRTC	Так – динамічна зміна бітрейту/якості залежно від фідбеку (вбудований congestion control)
QUIC	Ні (призначений для HTTP, не підтримує ABR самостійно)
MPTCP	Ні (керує декількома шляхами, але не змінює бітрейт)
DCCP	Ні (регулює швидкість через CCID, але без ABR)
UDP-Lite	Ні

Таблиця 2.6 — Масштабованість

Протокол	Масштабованість
TCP	середня – може використовувати CDN (HTTP), але кожне з’єднання окреме
UDP	середня – теоретично підтримує multicast, але в інтернеті – лише unicast
RTP	низька – точка–точка (потік до кожного клієнта)
RTSP	низька – точка–точка (в основному у відеоспостереженні)
RTMP	середня – сам по собі погано масштабується (великі аудиторії потребують CDN або ретрансляторів)
HLS	висока – легко масштабується через CDN (лінійно з кількістю серверів)
DASH	висока – так само через CDN (HTTP)
SRT	низька – точка–точка (для кожного глядача потрібен окремий потік або SFU)
WebRTC	низька – P2P/серверна модель (на велику кількість клієнтів потрібні SFU або MCU)
QUIC	висока – HTTP/3 працює через CDN (підтримка HTTP)
MPTCP	низька – точно двостороннє (два кінці), не для широкого розповсюдження
DCCP	низька – двостороннє, не для масових передач
UDP-Lite	середня – як UDP (можна multicast на локальній), але немає CDN

Таблиця 2.7 — Надійність доставки

Протокол	Надійність доставки
TCP	висока – гарантовано упорядкована і повторна передача втрач. пакетів
UDP	низька – немає гарантій доставки (аплікації шалують пакети самостійно)
RTP	низька – використовує UDP (немає гарант. доставки; втрати обробляються на рівні медіа)
RTSP	низька – потоки по суті ненадійні (при межі надійність залежить від протоколу доставки)
RTMP	висока – TCP, гарантує доставку без втрат
HLS	висока – HTTP/TCP, кожний сегмент доставляється надійно
DASH	висока – HTTP/TCP (як HLS)
SRT	висока – вбудований ARQ, мінімізує втрачені пакети (забезпечує відновлення)
WebRTC	висока – обов’язково застосовує DTLS/SRTP (всі канали зашифровані і захищені)
QUIC	висока – надійний (TLS 1.3 + повтор, немає HoL-блокування)
MPTCP	висока – як TCP (забезпечує доставку через кілька шляхів)
DCCP	низька – не гарантує доставку або порядок
UDP-Lite	низька – без гарант. доставки (лише часткова перевірка)

Таблиця 2.8 — Підтримка платформ

Протокол	Підтримка платформ
TCP	універсальна (підтримується всіма ОС та пристроями)
UDP	універсальна (усі ОС, широко використовується в мережах)
RTP	широка (VoIP-системи, медіасервери, IP-камери)
RTSP	широка (IP-камери, медіасервери, плейери)
RTMP	обмежена (необхідні спеціальні плеєри/сервери, Flash; зараз переважно RTMPS)
HLS	дуже широка (нативна підтримка в iOS/macOS; Android і браузери через MSE)

Продовження таблиці 2.8

DASH	широка (підтримка в багатьох браузерах та плеєрах; Android)
SRT	середня (бібліотеки/SDK для серверів і прикладних програм, не вбудований в ОС)
WebRTC	дуже широка (вбудовано в Chrome, Firefox, Edge, Safari та ін. браузери)
QUIC	широка (HTTP/3 підтримується у >95% сучасних браузерів)
MPTCP	обмежена (підтримка у Linux 5.6+, iOS, деякі роутери)
DCCP	обмежена (зазвичай підтримується в Linux/Unix, рідше вбудовано)
UDP-Lite	обмежена (Linux та деякі вбудовані системи)

Таблиця 2.9 — Безпека (рівень шифрування/захисту)

Протокол	Захист
TCP	немає вбудованого (можна використовувати TLS поверх TCP)
UDP	немає вбудованого (можна використовувати DTLS або IPsec)
RTP	нешифрований (опціонально SRTP – шифрування потоків)
RTSP	зазвичай нешифрований (є RTSPS/TLS-версія)
RTMP	нешифрований (RTMPS використовує TLS/SSL)
HLS	може використовувати AES-128 шифрування сегментів або HTTPS для доставки (Secure HLS)
DASH	підтримує CENC (AES-128) та HTTPS (захист через протоколи HTTP)
SRT	вбудоване AES-128/256 шифрування (захист відео)
WebRTC	завжди зашифрований (DTLS для сигналізації, SRTP для медіа)
QUIC	вбудований TLS 1.3 (повне шифрування з’єднання)
MPTCP	як TCP (можна надбудувати TLS)
DCCP	немає вбудованого (можна IPsec, але рідко використовується)
UDP-Lite	немає вбудованого

Аналіз сильних і слабких сторін протоколів

HTTP-протоколи HLS і MPEG-DASH ідеально підходять для масштабного поширення відеоконтенту завдяки роботі поверх TCP і широкій сумісності з CDN. Вони забезпечують надійну доставку сегментів і

шифрування (AES-128, CENC), але мають значну затримку (до 30 с). Навіть у режимах Low-Latency, ці протоколи не здатні забезпечити реальний час, тому не підходять для інтерактивних сценаріїв.

Натомість UDP-орієнтовані протоколи — WebRTC, SRT, RTP/RTSP — розроблені для мінімальної затримки та толерантності до втрат. WebRTC, з типовою затримкою до 0.5 с, підтримує наскрізне шифрування (DTLS/SRTP), браузерну інтеграцію і є основою відеоконференцій. SRT забезпечує надійну передачу високоякісного відео навіть у нестабільних мережах завдяки ARQ-механізмам і підтримці AES-шифрування. Проте для обох рішень масштабування вимагає серверної інфраструктури (наприклад, SFU).

RTMP, хоча й застарілий, досі використовується через низьку затримку (3–5 с) і стабільну доставку через TCP. Його використання обмежене потребою у Flash та відсутністю підтримки масштабування. RTP/RTSP зустрічаються в закритих рішеннях (IP-камери) і забезпечують мінімальні затримки, але не гарантують повну цілісність даних і не мають вбудованого механізму надійності.

Сучасні транспортні розширення, як-от QUIC (HTTP/3) і MPTCP, пропонують додаткові переваги: QUIC — це UDP-базований протокол з TLS 1.3, зниженою затримкою і відсутністю потреби в повторному встановленні з’єднання, а MPTCP дозволяє одночасно використовувати кілька мережевих інтерфейсів, покращуючи стабільність і пропускну здатність. Інші спеціалізовані рішення, як DCCP і UDP-Lite, підходять для специфічних сценаріїв — адаптивного стримінгу або нестабільних каналів зв’язку.

2.5 Виклики та обмеження у прямих трансляціях

Прямі відеотрансляції мають низку специфічних викликів, що виникають через особливості передачі даних у реальному часі та організаційні фактори. У цьому підпункті розглянуто головні технічні проблеми і бізнес-

обмеження стрімінгу, а також наведено приклади з практики YouTube Live, Twitch, Zoom та інших сервісів.

Технічні виклики

- Затримка передачі (network latency): Затримка передачі означає інтервал часу між передачею відеопотоку та його відтворенням у глядача. Вона зростає через відстань між серверами та клієнтами, завантаженість мережі й обмежену пропускну здатність каналів. Висока затримка робить неможливою злагоджену двосторонню комунікацію в реальному часі (наприклад, реакції глядачів чи синхронний переклад) і погіршує якість сприйняття трансляції. Користувачі помічають, що звук або реакції коментаторів запізнюються відносно відео, що знижує інтерактивність і задоволеність від перегляду.
- Джиттер і втрати пакетів: Джиттер — це нерівномірність у часі доставки мережевих пакетів. За високого джиттера кадри відео надходять із непередбачуваними затримками, в результаті з’являються перерви у відтворенні, пікселізація, артефакти або стрибки якості. Втрата пакетів ще більше ускладнює ситуацію, адже відбуваються повторні запити даних, що збільшує затримку та може спричинити повне зависання трансляції. У сумі ці явища викликають часті зависання і буферизацію навіть за стабільного початкового з’єднання, що помітно погіршує якість прямого ефіру.
- Нестабільна пропускна здатність мережі: Якщо доступна пропускна здатність мережі коливається (наприклад, у слабкому Інтернет-з’єднанні), система адаптивного стрімінгу змушена автоматично знижувати бітрейт чи збільшувати буфер для уникнення перебоїв. За низької пропускної здатності відео не може безперебійно відтворюватися, починаються затримки та буферизація. Навпаки, достатня швидкість Інтернету дозволяє забезпечити плавну, високо якісну трансляцію без зайвих затримок. Нестабільна швидкість інтернет-каналу є частою причиною зниження роздільної здатності стріму або раптових «стрибків» якості з метою зменшити навантаження.

- Обмеження мобільних/бездротових мереж: При стрімінгу через Wi-Fi, 3G/4G/5G або інші бездротові мережі з’являються додаткові ризики. Сигнал може спотворюватися перешкодами (стінами, іншими приладами), а базові станції легко перевантажуються великою кількістю користувачів. Наприклад, Wi-Fi часто відчуває деградацію сигналу через фізичні перешкоди, що спричиняє втрати пакетів і разові зупинки трансляції. У мобільних мережах при переміщенні між базовими станціями змінюється потужність зв’язку, що може призводити до короткочасних обривів чи значних коливань затримки. Загалом бездротові під’єднання менш стабільні, тому користувачі можуть бачити більше «зависань» і падінь якості стріму, ніж при дротовому доступі.
- Масштабованість при високому навантаженні: Під час великих подій (спортивних матчів, прем’єр, ігрових конференцій) до платформи одночасно підключається величезна кількість глядачів, що вимагає миттєвого масштабування ресурсів. Різке збільшення аудиторії може створювати сотні тисяч одночасних запитів, які треба обробити без збоїв. Наприклад, в квітні 2025 р. платформа Tubi організувала пряму трансляцію Суперкубка з піковим навантаженням 15,5 млн одночасних глядачів. Щоб витримати такий сплеск (до 166 667 запитів на секунду при стрімкому прирості 10 млн глядачів за хвилину), Tubi застосувала мульти-CDN стратегію, автоматичне масштабування серверів і реальний моніторинг трафіку. Без подібних заходів різкий стрибок аудиторії призвів би до збоїв у сервісі та довгих затримок для глядачів.

Організаційні та бізнес-виклики

- Ліцензування контенту (авторське право, DRM): Трансляція контенту без належних прав порушує авторські права. Потокові сервіси повинні укладати угоди з правовласниками, а також використовувати технології DRM для захисту відео. Наприклад, у YouTube Live в реальному часі сканують потоки на предмет захищеного контенту: якщо ефір містить чужий музичний чи відеоматеріал без ліцензії, трансляцію можуть призупинити або замінити на заставку. Використання DRM (на кшталт

Widevine, FairPlay, PlayReady) забезпечує захист правовласників, але ускладнює архітектуру системи і збільшує витрати на ліцензування та підтримку сумісності.

- Вартість CDN та інфраструктури: Доставка потоку широкому колу глядачів потребує розгалуженої CDN та потужних серверів, що дорого коштує. Згідно з аналізом ринку, витрати на CDN найчастіше складають найбільшу статтю витрат OTT-платформ. Постачальники мережевих послуг беруть оплату за обсяг переданих даних (звичайно за ТБ), тому довготривалі трансляції у високій якості швидко «з'їдають» бюджет. Крім CDN, значні витрати пов’язані з серверами кодування, зберігання та мережею: всі ці складові необхідні для безперебійного стрімінгу під час пікових навантажень.
- Забезпечення доступності в різних регіонах: Трансляції мають бути доступними для глядачів з різних країн, але територіальні обмеження часто ускладнюють це завдання. Зокрема, контент може бути ліцензований лише для певних регіонів, тому провайдери використовують гео-блокування. Наприклад, платформа може дозволяти перегляд стріму в США, але блокувати той самий вміст в Європі через міжнародні договори. Також деякі країни вимагають локалізації даних (наприклад, зберігання серверів у межах регіону), що примушує провайдерів розгорнути окрему інфраструктуру. Наявність чи відсутність CDN PoP (точок присутності) у віддалених регіонах також позначається на якості: там, де PoP немає, відео може йти з більшою затримкою або взагалі бути недоступним.
- Відповідність правовим нормам (GDPR, місцеве законодавство): Потоки часто збирають дані про глядачів (IP-адреси, поведінка, реєстраційні дані), які підпадають під захист законодавства про приватність. Платформи, що працюють в ЄС, мають строго дотримуватись GDPR. Це вимагає, зокрема, зберігати дані в зашифрованому вигляді, впроваджувати захищену автентифікацію користувачів і забезпечувати можливість вибору про обробку їхніх даних. CDN-партнери, які передають контент, також мають гарантувати захист персональних даних (IP-адреси і метаданих трафіку можуть

кваліфікуватися як персональні дані). Недотримання цих вимог може привести до великих штрафів і втрати репутації. Організатори трансляцій повинні чітко інформувати глядачів про збирання даних, надавати можливість їх видалення та забезпечувати зберігання інформації в межах встановлених регіональних норм.

Приклади з практики стрімінгових сервісів

- YouTube Live: У реальних трансляціях YouTube активно використовує автоматичний аналіз вмісту. Якщо прямий ефір містить сторонній авторський контент, система відразу сигналізує про порушення – трансляцію можуть поставити на паузу або перервати. Наприклад, музичні канали часто стикаються з тим, що пісні без ліцензії призводять до блокування ефіру. Окрім цього, YouTube Live намагається мінімізувати буферизацію, але за складних умов мережі глядачі все одно можуть помічати затримку або «плаваючий» FPS, особливо коли з’єднання користувача не стабільне.
- Twitch: На Twitch ігрові трансляції та кіберспортивні події приваблюють величезну аудиторію. Наприклад, на одній з трансляцій E3–2018 під час пресконференції Xbox одночасно дивилося близько 1,7 млн глядачів. Це встановило новий рекорд для платформи. Хоча сама служба встояла і не впала, обробка мільйонів одночасних з’єднань — серйозне технічне випробування. Twitch використовує режим низької затримки та буферний менеджмент, щоб унеможливити масову деградацію якості, але все одно деякі глядачі можуть фіксувати короткі паузи або нестабільність стріму, якщо їх інтернет-канал перевантажено чи нестабільний.
- Zoom: Навіть платформи відеоконференцій можуть зазнавати глобальних збоїв. Так, 16 квітня 2025 р. стався масштабний збій Zoom: через помилку на рівні DNS весь сервіс вийшов із ладу на кілька годин, і користувачі по всьому світу не могли приєднатися до зустрічей. Цей інцидент показав, як непередбачувані технічні несправності (навіть не пов’язані безпосередньо з відеопотоком) можуть повністю паралізувати роботу платформи та відключити живе спілкування учасників.

- Інші (наприклад, Netflix): Великі події, що передаються в режимі реального часу, іноді приводили до збоїв навіть на гігантських потокових сервісах. Наприклад, під час прямої трансляції боксерського поєдинку Джейка Пола та Майка Тайсона (листопад 2024) глядачі Netflix масово скаржилися на падіння якості відео: з’явилася пікселізація, «заїкання» картинки і навіть зупинки стріму. За словами самих користувачів і повідомлень у пресі, цьому сприяла рекордна кількість одночасних підключень — Netflix відзначила, що це було найбільш масове спортивне трансляційне шоу на їхній платформі. Подібні приклади демонструють, що високий попит і мережеві обмеження можуть зумовити технічні проблеми навіть у найпотужніших провайдерів.

2.6 Висновки з розділу 2

У другому розділі було здійснено ґрунтовний аналіз основних протоколів, що застосовуються у сфері мультимедійного мовлення в IP-мережах. Розглянуто архітектурні особливості, принципи функціонування, переваги та недоліки таких протоколів, як HLS, MPEG-DASH, RTMP, SRT, RTP/RTSP, WebRTC, а також перспективні транспортні рішення, включаючи QUIC та MPTCP.

HTTP-базовані протоколи (HLS, MPEG-DASH) забезпечують надійне, масштабоване та якісне доставлення контенту до широкої аудиторії завдяки сумісності з CDN, однак мають вищу затримку, що обмежує їх використання у сценаріях реального часу. Натомість протоколи, орієнтовані на мінімізацію затримок (WebRTC, SRT), демонструють високу ефективність у відеоконференціях, інтерактивних стрімах та професійному мовленні, хоча потребують складнішої інфраструктури для масштабування.

Також було охарактеризовано сучасні тенденції розвитку транспортних протоколів, що покликані покращити адаптацію до мобільних та нестабільних мереж (QUIC, MPTCP), а також спеціалізовані рішення для умов із частковими втратами (DCCP, UDP-Lite).

Таким чином, вибір конкретного протоколу чи комбінації рішень має базуватися на вимогах до затримки, надійності, масштабованості, безпеки та типу взаємодії (однонаправлений стримінг чи двосторонній зв’язок). Універсального рішення не існує — кожна технологія оптимізована під свій клас задач, що й визначає її ефективність у практичному використанні.

3 ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОТОКОЛІВ ПЕРЕДАЧІ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

3.1 Критерії оцінювання ефективності протоколів

У сфері прямих мультимедійних трансляцій ефективність протоколу визначається насамперед параметрами, що безпосередньо впливають на якість сприйняття аудиторією (QoE) та на технічні показники якості обслуговування (QoS). До ключових критеріїв належать затримка передачі, джиттер (нестабільність доставки пакетів), втрата пакетів, адаптивність до зміни пропускної здатності мережі, масштабованість, надійність доставки, кінцева якість зображення і звуку, витрати ресурсів, а також безпека і сумісність із різними платформами. Кожний із цих параметрів важливий у контексті прямих трансляцій, оскільки безпосередньо впливає на досвід глядача. Нижче розглянемо кожен критерій окремо, наведемо його визначення, пояснимо значимість саме для live-відтворення та приклади впливу на користувацький досвід.

Затримка (latency) — це проміжок часу між моментом, коли мультимедійний пакет залишає джерело, та моментом, коли він досягає свого призначення. Цей показник є надзвичайно критичним для додатків реального часу, таких як відеоконференції, онлайн-ігри або трансляції у прямому ефірі. Затримка складається з декількох компонентів: час кодування, передача по мережі, маршрутизація, декодування та рендеринг. У системах, що базуються на протоколах HTTP, таких як HLS або MPEG-DASH, затримка може перевищувати 20–30 секунд через потребу у попередньому формуванні сегментів відео, буферизацію на стороні клієнта та кешування в CDN. Ці значення є неприйнятними для інтерактивних сервісів. Для зниження затримки використовують UDP-базовані протоколи — WebRTC, SRT, RTP, які можуть забезпечити доставку з затримкою менш ніж 500 мілісекунд. У деяких сценаріях використання спеціалізованих серверів (наприклад, SFU або Media Relay) дозволяє оптимізувати топологію доставки та ще більше скоротити

затримку. Додатково до цього важливими є параметри мережевої інфраструктури: якісні маршрутизатори, QoS-пріоритезація потоків, мінімізація кількості переходів між вузлами — усе це дозволяє скоротити затримку до мінімальних значень.

Джиттер (jitter) — це варіабельність затримки, тобто флуктуація часу прибуття окремих пакетів. Для мультимедійного потоку рівномірність надходження пакетів є критичною: коли один кадр відео приходить через 30 мс, а наступний — через 150 мс, виникають артефакти, ривки або розсинхронізація аудіо і відео. Джиттер виникає з різних причин: зміни маршруту, перевантаження буферів на мережевих пристроях, коливання пропускної здатності каналу. Для згладжування джиттера зазвичай застосовуються буфери, але вони автоматично збільшують загальну затримку. WebRTC має вбудовані алгоритми адаптації, які аналізують RTCP-звіти, визначають тенденції у мережевій поведінці та підлаштовують параметри потоку. У свою чергу, SRT дозволяє налаштовувати розмір буфера затримки вручну, що дає більший контроль над компромісом між стабільністю відтворення і загальною затримкою. Деякі пристрої підтримують апаратне згладжування джиттера, що також покращує відтворення.

Втрати пакетів — одна з основних проблем у передаванні мультимедіа через ненадійні мережі, зокрема Wi-Fi, мобільні або супутникові з’єднання. Коли пакет втрачається, це призводить до зникнення окремих кадрів, затримок, розмиття зображення або повного припинення потоку. Протоколи TCP гарантують повторну доставку втрачених пакетів, але це неминуче викликає затримки. Для реального часу прийнятним рівнем вважаються втрати до 1–2%, а в мобільних мережах — до 5%. UDP-протоколи мають власні механізми пом’якшення наслідків втрат. WebRTC застосовує техніки FEC, NACK та PLI. FEC додає надлишкову інформацію, яка дозволяє відновлювати втрачені дані без повторного запиту. NACK запитує повторну доставку лише для втрачених кадрів, не блокуючи загальний потік. У SRT реалізовано ARQ з детальним управлінням таймінгом запитів. Додатково

можна використовувати резервні канали, багатошляхову маршрутизацію (MPTCP) або технології типу bonded streaming.

Адаптивність до мережі — властивість системи динамічно змінювати параметри потоку залежно від умов середовища. В реальних мережах пропускна здатність постійно змінюється — через зміну навантаження, перемикання між базовими станціями (handover) або інші фактори. HTTP-протоколи мають адаптацію на рівні сегментів — гравець перемикається між якістю 240p/480p/720p/1080p. Але перемикання відбувається із затримкою у кілька секунд, що робить цей механізм неефективним у реальному часі. Натомість WebRTC використовує транспортний контроль перевантаження (TCC), Congestion Control та Real-Time Bandwidth Estimation, які дозволяють змінювати бітрейт, FPS, роздільну здатність у реальному часі. Це забезпечує безперервність потоку навіть при різкому зниженні швидкості каналу. SRT теж підтримує адаптацію: шляхом аналізу RTT, packet loss та jitter можна коригувати швидкість передачі і розмір буфера.

Масштабованість (scalability) — відображає здатність мережевої системи обслуговувати зростаючу кількість клієнтів без суттєвої втрати якості або збільшення затримок. Для пасивного мовлення ідеальними є HLS/DASH — завдяки CDN вони масштабуються горизонтально і практично необмежено. Для WebRTC ситуація складніша: кожен новий клієнт — новий поток. Для цього застосовують MCU (Multipoint Control Unit) або SFU, які зменшують навантаження на клієнтів. SFU передає кожному клієнту тільки потрібний потік. Це дозволяє масштабувати до 50–100 одночасних користувачів. Для ще більшого навантаження застосовують каскадні SFU, хмарні інфраструктури (AWS MediaConnect, Google Cloud Media), або комбіновані моделі — WebRTC для активних учасників, HLS для глядачів.

Надійність доставки — залежить від здатності системи гарантувати передачу всіх або максимальної кількості пакетів без порушення структури даних. У професійному відеомовленні критично, щоб навіть при втраті частини мережі (наприклад, один з каналів зв'язку вийшов з ладу) потік

залишався живим. Для цього застосовують дублювання потоку на кілька інтерфейсів, резервні маршрути, протоколи з самовідновленням. У UDP-протоколах, таких як SRT, реалізована логіка повторного запиту з урахуванням допустимої затримки. У WebRTC впроваджено гнучку політику компенсації втрат: FIR, REMB, підтримка redundancy RTP. Деякі системи комбінують RTP та FEC для додаткового захисту.

Якість зображення (video quality) — залежить від параметрів кодування (кодек, бітрейт, роздільна здатність), стану мережі, типу адаптації та наявності механізмів компенсації втрат. Сучасні кодеки (VP9, AV1) дозволяють при однаковому бітрейті отримати вищу якість у порівнянні з H.264. WebRTC постійно змінює параметри в залежності від стану каналу, дозволяючи зберігати стабільність зображення. SRT підтримує високу якість при стабільному каналі, і використовується в студійних трансляціях. HTTP-протоколи (HLS/DASH) покладаються на заготовлені профілі якості — чим більше профілів, тим вища гнучкість. Важливо також зберігати кольорові профілі (BT.709, BT.2020), що є важливим для передачі HDR-контенту.

Ресурсозатратність (system load) — параметр, що оцінює споживання обчислювальних і енергетичних ресурсів системою трансляції. У мобільних пристроях важливими є оптимізація використання CPU та економія заряду батареї. WebRTC вимагає багато ресурсів — обробка кількох відеопотоків, шифрування, адаптація — усе це лягає на CPU. У десктопах або серверах ці витрати менш критичні. SRT — більш легкий, але вимагає додаткового програмного забезпечення (наприклад, Haivision або OBS з підтримкою SRT). HLS і DASH переважно споживають пам’ять (буферизація), але навантаження на процесор невелике. У великих трансляціях застосовують апаратні енкодери (NVENC, QuickSync), що значно знижують загальну ресурсозатратність.

Безпека (security) — забезпечення конфіденційності, цілісності та доступності потоку. WebRTC — найбезпечніший з точки зору базової архітектури: усі потоки шифруються DTLS/SRTP, а доступ до камер і мікрофонів суворо регламентується браузером. SRT має підтримку AES-

шифрування, автентифікацію джерел, і дозволяє вбудовувати DRM. HLS та DASH можуть передаватися через HTTPS, а також підтримують індустріальні DRM: Widevine, PlayReady, FairPlay. У хмарних рішеннях також реалізується управління ключами, інтеграція з IAM (Identity and Access Management), а також моніторинг підозрілої активності. Безпечна трансляція — ключ до захисту авторських прав та даних користувачів.

Сумісність (compatibility) — фактор, який визначає здатність технології працювати на різних платформах і пристроях. WebRTC підтримується всіма сучасними браузерами (Chrome, Firefox, Edge, Safari), має SDK для Android та iOS. SRT активно інтегрується в професійні системи (OBS, vMix, Wowza). HLS — стандарт для Apple, MPEG-DASH — для Android. QUIC та HTTP/3 — майбутнє потокового відео, що дозволяє уникати затримок на етапі встановлення з'єднання та ефективніше використовувати пропускну здатність. Системи повинні підтримувати fallback — наприклад, якщо WebRTC не підтримується, автоматично перемикатися на HLS.

3.2 Методологія аналітичного порівняння

Методологія аналітичного порівняння протоколів передачі мультимедійного контенту передбачає систематичну оцінку їх ефективності за сукупністю критеріїв, сценаріїв використання та практичних вимог. Такий підхід дає змогу не лише технічно зіставити окремі рішення, а й зробити обґрунтований вибір відповідно до реальних умов експлуатації. У сфері мультимедійного мовлення важливими є не лише технічні характеристики, а й поведінка протоколів в умовах змішаного трафіку, різного рівня завантаження мережі, втрат чи затримок. Тому вибір методології повинен бути достатньо гнучким, щоб охоплювати як кількісні, так і якісні параметри.

Ключовими етапами аналітичного порівняння є: визначення об'єктивних критеріїв ефективності, формулювання типових сценаріїв використання, збір даних із технічної документації та практичних кейсів, а також узагальнення результатів за допомогою описових і табличних методів.

Всі ці елементи формують комплексну базу для ухвалення технічних рішень при виборі протоколу передачі мультимедійного контенту.

У рамках цієї методології виділяється низка критеріїв порівняння, які дозволяють охарактеризувати сильні та слабкі сторони протоколів. Ці критерії охоплюють технічні, мережеві, якісні та безпекові аспекти, що є ключовими для забезпечення безперервної, якісної й захищеної передачі аудіо- та відеоданих.

1. Затримка передачі (Latency): це сукупний час, необхідний для того, щоб мультимедійні дані пройшли від джерела до кінцевого споживача. Цей параметр є критичним для всіх застосунків реального часу — зокрема, для відеоконференцій, онлайн-ігор, кіберспортивних трансляцій. Протоколи на базі UDP зазвичай мають мінімальні затримки, однак не гарантують доставку. Натомість TCP, хоч і забезпечує впорядковану передачу та надійність, затримує трафік через підтвердження й можливість повторної передачі. Протокол QUIC, який є основою HTTP/3, створений для мінімізації затримок завдяки відсутності багатоетапного встановлення з’єднання, що є перевагою для адаптивних стрімінгів.

2. Втрати пакетів (Packet loss): у мультимедійній передачі навіть малі втрати можуть призводити до розривів звуку чи візуальних артефактів. У реальному часі протоколи повинні або запобігати втратам, або мати засоби для їх компенсації. Наприклад, RTP має механізми нумерації кадрів і компенсації джиттеру. Протокол SRT, розроблений спеціально для відео у нестабільних мережах, може витримувати до 10% втрат без помітного зниження якості. У деяких випадках втрата пакету менш критична, ніж затримка повторної передачі, тож протокол повинен балансувати між швидкістю й цілісністю.

3. Адаптивність: здатність протоколу динамічно реагувати на зміну пропускної здатності каналу є критичною при мобільному доступі або нестабільному Wi-Fi. Адаптивні HTTP-протоколи, зокрема HLS і MPEG-DASH, реалізують цей принцип шляхом розбиття відео на сегменти різного бітрейту. Клієнт визначає свою пропускну здатність і автоматично

перемикається на сегмент з відповідною якістю, зменшуючи буферизацію й забезпечуючи безперервність відтворення. Такі протоколи широко підтримуються мобільними пристроями, браузерами та CDN-інфраструктурою.

4. Масштабованість: під цим критерієм розуміється здатність протоколу підтримувати ефективну роботу при зростанні кількості одночасних користувачів або потоків. Протоколи на основі HTTP (HLS, DASH) легко масштабуються завдяки використанню CDN і кешування — контент подається через звичайні веб-сервери, що зменшує навантаження на джерело. RTP підтримує IP-мультикаст, що дозволяє розсилати один потік на багато одержувачів одночасно, проте вимагає відповідної мережевої підтримки. Натомість WebRTC, працюючи через пірингові або SFU-архітектури, має обмеження щодо кількості одночасних потоків і менш придатний для масштабного мовлення без зовнішніх сервісів.

5. Надійність: визначається тим, наскільки точно і повно протокол забезпечує доставку даних до отримувача. TCP — приклад протоколу з гарантованою доставкою, включаючи механізми контролю порядку і повторної передачі. UDP, навпаки, працює без підтверджень, але завдяки цьому зменшує затримки. Компромісні варіанти, як-от SRT або DCCP, намагаються поєднати високу швидкість з корекцією помилок та адаптивним контролем навантаження. Важливо враховувати, що у мультимедіа не завжди потрібна 100% надійність, якщо невеликі втрати не псують загальне враження.

6. Ресурсоємність: кожен протокол вимагає певного обсягу обчислювальних ресурсів — пам’яті, процесорного часу, пропускної здатності. UDP має мінімальні накладні витрати, оскільки не містить механізмів контролю, тоді як TCP, QUIC чи SRT мають складнішу структуру заголовків і функції повторної передачі. Адаптивні протоколи, як HLS/DASH, потребують додаткового кодування відео на кілька потоків різного бітрейту, що збільшує навантаження на серверну частину.

7. Безпека: сучасні протоколи включають засоби шифрування й автентифікації. Наприклад, RTMP має зашифровану версію RTMPS, HLS і DASH можуть працювати через HTTPS з AES-шифруванням. WebRTC передбачає наскрізне шифрування аудіо та відео. Протокол SRT також підтримує AES-шифрування, що робить його придатним для захищених трансляцій.

8. Сумісність: важливо, щоб протокол безперешкодно проходив через мережеві фаєрволи, проксі та NAT. HTTP-протоколи (порт 80/443) проходять без обмежень у більшості середовищ, тоді як UDP-орієнтовані рішення можуть бути заблоковані. Саме тому WebRTC, який застосовує ICE/STUN/TURN, і HLS/DASH отримали широку підтримку в браузерах і мобільних клієнтах.

Усі наведені критерії становлять основу подальшого аналізу та порівняння, де їх значення розкривається у прикладних умовах — залежно від типу трансляції, аудиторії та технічних ресурсів платформи.

Типові сценарії використання протоколів

Для повноцінного аналізу недостатньо лише технічних характеристик — важливо враховувати й прикладні сценарії, у яких використовуються протоколи. Різні види мультимедійного контенту вимагають від протоколів різного набору властивостей. Розглянемо основні з них.

1. Масові трансляції: Такі платформи, як YouTube Live, Facebook Live чи Twitch, передають один потік на мільйони глядачів. У таких випадках головними критеріями стають масштабованість і сумісність. Протоколи, що використовують CDN (наприклад, HLS, MPEG-DASH), дозволяють ефективно розподіляти навантаження, забезпечуючи безперервність відтворення. Стандартна схема включає RTMP або SRT для інгесту потоку (від мовника до сервера), після чого відео транскодується й розповсюджується через HTTP-протоколи.

2. Відеоконференції та інтерактивне спілкування: У таких сценаріях, як Zoom, Google Meet чи Skype, затримка повинна бути мінімальною. Користувач

очікує практично миттєву реакцію співрозмовника. Протоколи WebRTC або RTP, що працюють поверх UDP, оптимальні для цього: вони мінімізують затримку і дозволяють пірингове з’єднання з вбудованим шифруванням. Хоча ці протоколи гірше масштабуються, для конференцій із десятками або навіть сотнями учасників вони залишаються ефективними.

3. Мобільний доступ: Користувачі смартфонів або планшетів зазвичай підключаються через нестабільні канали, як-от Wi-Fi, 3G або LTE. У таких випадках важлива адаптивність — можливість зміни якості потоку залежно від пропускної здатності. HLS і MPEG-DASH дозволяють динамічно перемикати якість сегментів, що знижує ризик буферизації. Ці протоколи також добре працюють із проксі, фаєрволами і є сумісними з більшістю мобільних браузерів.

Таким чином, кожен сценарій висуває власні вимоги до протоколів. У подальшому аналізі їх ефективність буде оцінено з урахуванням специфіки практичного застосування.

Збір даних для порівняльного аналізу

Щоб провести об'єктивне порівняння протоколів, необхідно спиратися на достовірні джерела даних. Вони включають як формальні специфікації, так і практичні результати вимірювань, що дозволяють оцінити реальну ефективність протоколів у конкретних умовах. Основними джерелами для збору даних є:

1. Технічна документація та стандарти: Офіційні документи, що описують структуру та функціональні можливості протоколів. Наприклад, RTP детально описано в RFC 3550, HLS — в RFC 8216, а MPEG-DASH — у міжнародному стандарті ISO/IEC 23009-1. Також важливими є сучасні специфікації таких протоколів, як QUIC (RFC 9000) чи DCCP (RFC 4340). Ці документи містять ключову інформацію про можливості, обмеження і призначення кожного протоколу.

2. Наукові дослідження: Академічні статті та дослідницькі роботи дозволяють оцінити протоколи в реальних або модельованих умовах. У них

часто подаються результати тестування затримки, втрат пакетів, адаптивності або якості відео в різних мережах. Такі дані є корисними при визначенні практичних переваг одного протоколу над іншим у визначених сценаріях (наприклад, порівняння SRT і WebRTC для low-latency стрімінгу).

3. Публікації технічних компаній: Технічні блоги та white papers від таких компаній, як Google, Meta, Haivision, Amazon Web Services, надають інформацію про реальне впровадження та використання протоколів на платформах YouTube Live, Facebook Live, Twitch, Zoom тощо. Наприклад, у блозі Google описуються компроміси між використанням RTMP і HLS для трансляцій, а Haivision публікує кейси застосування SRT у професійних студіях.

4. Практичні кейси та вимірювання: До цієї категорії належать дані, отримані з робочих систем — зокрема, затримки при використанні HLS із різними розмірами сегментів, показники втрат при WebRTC-конференціях або продуктивність протоколів у CDN-мережах. Такі дані мають особливу цінність, оскільки демонструють поведінку протоколів у продуктивних, а не лабораторних умовах.

Усі ці джерела формують інформаційну базу для подальшого аналізу. Комбінуючи технічні специфікації з реальними вимірами, можна отримати більш повну картину щодо ефективності та придатності протоколів до конкретних завдань.

Методи аналітичного порівняння

Щоб провести глибоке й об’єктивне порівняння протоколів, використовуються різні методи аналізу, які дозволяють оцінити їх властивості як кількісно, так і якісно. Найбільш поширеними серед них є:

1. Табличний метод: Передбачає створення узагальненої таблиці, в якій усі протоколи оцінюються за обраними критеріями — затримка, втрати, адаптивність, масштабованість, безпека тощо. Таблиця дозволяє наочно порівнювати протоколи між собою та швидко ідентифікувати сильні та слабкі

сторони кожного. Наприклад, можна побачити, що WebRTC має найменшу затримку, але поступається масштабованістю HLS/DASH.

2. Описовий аналіз: У цьому методі кожен протокол розглядається окремо, з акцентом на його ключові характеристики в контексті практичного використання. Наприклад, SRT детально описується як рішення для професійних трансляцій із високими вимогами до якості й надійності, тоді як HLS позиціонується як універсальний варіант для веб-стрімінгу. Такий підхід дозволяє врахувати нюанси, які не завжди відображені у таблицях.

3. Кейсовий аналіз: Цей метод базується на аналізі реальних прикладів використання протоколів у конкретних платформах або продуктах. Наприклад, у випадку YouTube Live можна простежити, як відбувається обробка потоку від моменту надсилання (RTMP) до кінцевого користувача (через HLS/DASH). Аналіз таких кейсів допомагає перевірити, як теоретичні переваги протоколів проявляються на практиці.

4. Аналітичний (квантифікативний) аналіз: Передбачає використання чисельних методів — моделювання, вимірювання, симуляції — для оцінки параметрів протоколів. Це можуть бути графіки затримок при збільшенні трафіку, статистика втрат пакетів у різних мережевих умовах або ефективність адаптивних алгоритмів при коливаннях бітрейту. Цей підхід забезпечує глибоке технічне розуміння роботи протоколів.

Застосування кількох методів одночасно дає змогу отримати комплексну й збалансовану картину, що особливо важливо в умовах, коли протоколи повинні відповідати багатьом, іноді взаємовиключним, вимогам.

Обґрунтування вибору протоколів для порівняння

Для аналітичного дослідження було обрано набір протоколів, які охоплюють основні підходи до передачі мультимедійного контенту в сучасних мережах. Вони відрізняються за архітектурою, принципами роботи, підтримкою надійності, адаптивності та масштабованості, що дозволяє сформувати повноцінну порівняльну картину.

TCP (Transmission Control Protocol): базовий транспортний протокол, що гарантує надійність і впорядкованість доставки. Активно використовується як основа для протоколів RTMP, RTMPS та HTTP-стрімінгів. Основний недолік — затримки, що виникають через механізми контролю потоку й повторної передачі.

UDP (User Datagram Protocol): протокол з мінімальною затримкою, який не передбачає контролю доставки. Він лежить в основі RTP, WebRTC, SRT і дозволяє швидко передавати мультимедійний трафік, хоч і з можливими втратами.

RTP (Real-time Transport Protocol): стандарт для передачі аудіо- й відео у реальному часі. Застосовується в VoIP, відеоконференціях та інтерактивних сервісах. Працює поверх UDP і має механізми синхронізації та компенсації втрат.

RTMP (Real-Time Messaging Protocol): протокол від Adobe, заснований на TCP. Досі активно використовується для інгесту потоків, попри застарілість технології Flash. Сумісний із багатьма програмами для стрімінгу.

HLS (HTTP Live Streaming): протокол адаптивного стрімінгу, розроблений Apple. Забезпечує гнучкість якості завдяки розбиттю потоку на сегменти. Має широку підтримку на мобільних пристроях та у браузерах.

MPEG-DASH: відкритий стандарт адаптивного стрімінгу, схожий за принципом на HLS, але незалежний від платформи. Підтримує сучасні кодеки й дає змогу масштабовано доставляти контент через CDN.

SRT (Secure Reliable Transport): сучасний протокол, який поєднує переваги UDP з механізмами надійності (ARQ, шифрування). Створений для професійних трансляцій в умовах нестабільних мереж.

WebRTC: комплекс веб-стандартів для прямої передачі медіа у браузері. Має дуже низьку затримку й високу безпеку, але обмежену масштабованість без використання SFU-серверів.

QUIC (Quick UDP Internet Connections): новітній транспортний протокол від Google, що став основою для HTTP/3. Забезпечує зменшену затримку завдяки уникненню повторного встановлення з’єднань.

DCCP та UDP-Lite: експериментальні протоколи, які дозволяють балансувати між швидкістю й надійністю (DCCP) або передавати частково пошкоджені пакети (UDP-Lite), що може бути корисно для відео з допуском до незначних спотворень.

Усі ці протоколи репрезентують різні підходи до мультимедійного стрімінгу — від класичних TCP-заснованих до сучасних UDP-адаптивних. Їх порівняння дозволяє охопити широкий спектр застосувань: від масових трансляцій і відеоконференцій до low-latency стрімінгів у кіберспорті чи мобільному середовищі.

3.3 Аналіз традиційних протоколів (TCP, UDP)

TCP та UDP – два основні транспортні протоколи стеку IP, що працюють на 4-му (транспортному) рівні OSI. TCP – орієнтований на з’єднання протокол: перед передачею даних відправник і отримувач узгоджують параметри зв’язку через трьохстороннє «рукостискання» (SYN → SYN-ACK → ACK), гарантують доставку, впорядковують пакети та керують потоком даних. Навпаки, UDP – безз’єднанковий (датиґрамний) протокол: повідомлення передаються без встановлення зв’язку, без підтверджень та повторних передач.

- Установка з’єднання (TCP): виконується трьохстороннім «рукостисканням».

1. SYN – клієнт надсилає сегмент з флагом SYN для ініціалізації зв’язку.
2. SYN-ACK – сервер відповідає, надсилаючи SYN+ACK (підтвердження отримання SYN).

3. ACK – клієнт завершує процес, відправляючи ACK. Після цього канал встановлено, і починається передача даних.

- Закриття з’єднання (TCP): використовується чотиристороннє роз’єднання. Зазвичай сторона, що завершує, надсилає FIN; друга сторони підтверджує ACK, потім йде FIN від другої сторони і заключне ACK першої сторони. Такий процес гарантує коректне завершення сеансу без втрати даних.
- Передача даних (TCP): дані розбиваються на сегменти. Кожен сегмент несе 20–60 байт заголовку (залежно від опцій) і поле даних. Заголовок містить номери портів (16 біт), номер послідовності, поле підтвердження, флаги (SYN, ACK, FIN тощо), розмір вікна приймача, контрольну суму, тощо. TCP забезпечує надійну доставку: отримані сегменти нумеруються за послідовністю і, у разі їх втрати чи пошкодження, відправник повторно відправляє пропущені фрагменти. Також TCP виконує контроль потоку (через протокол ковзного вікна) і контроль затримки (алгоритми slow start, congestion avoidance тощо), щоб уникати перевантаження каналу.
- Передача даних (UDP): протокол без установлення з’єднання передає окремі датиґрами. Заголовок UDP фіксованого розміру 8 байт містить лише два поля портів (по 16 біт), довжину пакета і контрольну суму. UDP не забезпечує перевірки порядку, підтверджень чи повторних передач – кожен датаграм обробляється незалежно. Це значно зменшує затримку й накладні витрати протоколу. UDP не гарантує доставку або впорядкування даних, що робить його «легким» і швидким, але непридатним для сценаріїв, де критична кожна втрата пакета.

Порівняння переваг/недоліків. TCP зазвичай забезпечує максимальну точність передачі: через ретельну перевірку помилок, перенумерацію та підтвердження він гарантує доставку всіх сегментів у правильному порядку. Наприклад, Wowza відзначає, що TCP «вимагає більш частої взаємодії пристроїв для перевірки помилок», тож якщо пакети приходять не у порядку чи з помилками, протокол вимагатиме повторення. Така надійність пояснюється механізмами підтверджень (ACK) і повторних передач. Водночас

ця надійність створює затримки: переважання підтверджень і повторень може викликати «довготривалі затримки (кілька секунд)» на очікування повторних передач. У Wowza констатують: «UDP однозначно перемагає в швидкості» – пакети UDP передаються без очікування retransmission, тому затримки мінімальні.

UDP, навпаки, із самого початку проектувався для мінімального затримування. Він не реалізує ніяких механізмів повторного передавання чи підтверджень, що зменшує затримку доставки і робить його ідеальним для реального часу. Прямий приклад – мультимедіа: UDP «підтримує низьку затримку та забезпечує плавне відтворення, навіть якщо деякі пакети загубляться». Завдяки відсутності стану зв'язку UDP легко масштабується – він «підтримує мультікаст і підходить для передачі потокового відео та опитування великої кількості клієнтів». Недоліки UDP очевидні: будь-яка втрата пакета чи «рвані» пакети призводять до пропуску кадрів або зривів аудіо, але в багатьох реальних потокових сервісах це менше шкодить відчуттю користувача, ніж затримки.

Сценарії використання в мультимедіа. У стрімінгу відео часто застосовують обидва підходи. TCP (наприклад через HTTP або RTMP) забезпечує надійну доставку кадрів і часто використовується для адаптивних протоколів на основі HTTP. Так, протокол RTMP (для живого стрімінгу) суворо ґрунтується на TCP. Так само HTTP/2 працює поверх TCP (загальний HTTP-стек), а HTTP/3 використовує QUIC поверх UDP. UDP же лежить в основі більшості real-time протоколів: RTP для передачі аудіо/відео, WebRTC (переважно через SRTP поверх UDP) і нові протоколи як SRT (Secure Reliable Transport) спроектовані на UDP з власним механізмом відновлення втрат. RTSP як протокол керування потоком може працювати і через TCP, і через UDP залежно від кейсу. Отже, для мультимедіа: наприклад, RTMP і старіші HLS/DASH (з HTTP) використовують TCP, а WebRTC/ RTP – UDP.

Параметри ефективності. За затримкою (latency) UDP традиційно швидший – відсутність очікування перекодовування чи підтверджень дозволяє

даним доставлятися максимально оперативно. З іншого боку, TCP знижує втрати (через ACK і повтори) і впорядковує дані, але за рахунок створення накладних витрат (зайві бітові поля, обмін рукостисканням, підтвердженнями). У Wooza підрахували, що заголовок TCP мінімум у 2.5 рази більше за UDP (20 vs 8 байт) і що TCP «важкий» через три пакети для встановлення з'єднання, тоді як UDP «легкий» та безстанний. Це означає, що TCP споживає більше пам'яті і CPU на підтримку з'єднань, але його пропускна здатність надійно використовується; UDP дозволяє працювати з набагато більшим числом клієнтів (через multicast/широкомовлення), але може втрачати пакети при великому навантаженні. Відтак, масштабованість UDP вищі (через безстанову архітектуру), а у TCP необхідно відслідковувати кожне з'єднання. Аналогічно, ресурсомісткість TCP більша (керування станом, алгоритми контролю), UDP – мінімальна.

Приклади платформ. У практичних застосунках це проявляється так. Наприклад, YouTube Live та Twitch використовують TCP-потоки (RTMP/HTTP). Twitch підтверджує це у документації: «інструмент передає відеосигнал на Twitch через RTMP» (тобто TCP). Аналіз трафіку YouTube показує, що платформа явно використовує TCP (щоб скористатися його алгоритмами slow start, retransmission тощо). Водночас Zoom (як і більшість відеоконференцій) передає медіа через UDP: у правилах Zoom вказано порти UDP 3478, 3479, 8801-8810 для медіапотоків. Наприклад, при з'єднанні Zoom намагається використовувати UDP для швидкої передачі аудіо/відео (TCP – здебільшого лише для контролю та сигналізації). OBS Studio – популярний стрімінговий софт – теж посилає потік у сервіси через RTMP (TCP). Таким чином, у цікавих випадках застосування: стрімінг контенту (YouTube Live, Twitch) відбувається переважно через TCP-канали, а інтерактивні відео-дзвінки (Zoom, WebRTC) – через UDP.

Таблиця 3.1 – Порівняльна характеристика транспортних протоколів TCP і UDP

Характеристика	TCP	UDP
Зв’язок	Орієнтований на з’єднання	Без встановлення з’єднання
Надійність доставки	Гарантована (перевірка, ACK, повтори)	Негарантована (пакети можуть бути втрачені)
Порядок доставки	Упорядкований (позиційні номери сегментів)	Неупорядкований (немає відстеження порядку)
Контроль потоку	Є (ковзне вікно)	Немає
Контроль перевантаження	Є (алгоритми slow start, congestion avoidance)	Немає
Установка з’єднання	3-стороннє «рукостискання» (SYN/SYN-ACK/ACK)	Немає (додаткових пакетів)
Закриття з’єднання	4-стороннє (обмін FIN/ACK)	Немає
Розмір заголовка	20–60 байт (мінімум 20)	8 байт (фіксований)
Накладні витрати	Високі (керування з’єднанням, підтвердження)	Низькі (немає служби підтверджень)
Затримка	Часто вища (через очікування ACK/повтор)	Зазвичай нижча (пряма відправка без повторів)
Масштабованість	Обмежена (статус кожного з’єднання)	Висока (безстановість, підтримка multicast)
Ресурсозатратність	Більше (стан, алгоритми)	Менше (мінімальна обробка)
Застосування (приклади)	HTTP/HTTPS, FTP, SMTP, передачі файлів, OBS (RTMP)	VoIP/відео (RTP, WebRTC), ігри, DNS

Аналітичні висновки. TCP є вибором, коли важлива надійність і впорядкованість – наприклад, звичайний веб (HTTP/HTTPS), передача файлів чи будь-який контент, де критично уникнути втрати даних. Однак це досягається ціною більших затримок і перевитрат на контроль (через SYN/ACK, підтвердження та повтори). UDP, навпаки, мінімізує затримку і накладні витрати – це вирішення для мультимедійних застосунків реального часу (відеодзвінків, ігор, IP-телебачення). У табличці вище зведено їхні

ключові відмінності. Таким чином, у мультимедіа протоколи вибираються залежно від пріоритету: стрімінгові платформи (YouTube, Twitch) традиційно працюють поверх TCP, а системи відеоконференцій і онлайн-ігор – на основі UDP.

3.4 Порівняння сучасних мультимедійних протоколів

Розглянемо сучасні мультимедійні протоколи, структуровано за чотирма категоріями: протоколи реального часу (RTP і WebRTC), протоколи потокового мовлення (RTMP і SRT), адаптивні HTTP-протоколи (HLS і DASH) та альтернативні транспортні протоколи (QUIC, DCCP, UDP-Lite). Для кожної групи наведено огляд ключових характеристик за такими критеріями: затримка, втрати, адаптивність, масштабованість, якість передачі, безпека, сумісність, ресурсоємність. Також наведено приклади застосувань, оцінку сильних і слабких сторін, порівняльні таблиці та аналітичні висновки.

1. Протоколи реального часу: RTP і WebRTC

RTP (Real-time Transport Protocol) – стандартний мережевий протокол для передачі аудіо- та відеопотоків у режимі реального часу. Використовується переважно поверх UDP без гарантій доставки, забезпечуючи низьку затримку. RTP підтримує нумерацію пакетів і відстеження часу, що дозволяє отримувачеві відтворювати мультимедіа у правильному порядку і синхронізовано. Часто RTP застосовується разом з RTCP (контрольний протокол) для зворотнього зв’язку і моніторингу якості. Протокол призначений для сценаріїв, де деяка втрата пакетів допустима, але критична мінімальна затримка (наприклад, IP-камери, VoIP, конференцзв’язок). RTP сам по собі не шифрується, але існує розширення SRTP (Secure RTP) для конфіденційності й аутентифікації.

WebRTC (Web Real-Time Communication) – відкрита технологія та набір протоколів для реального часового спілкування в браузерах і додатках. WebRTC використовує RTP/RTCP (зазвичай у формі SRTP) для передачі

аудіо/відео, а також стандарти ICE/STUN/TURN для встановлення зв'язку через NAT. Головна мета WebRTC – забезпечити майже миттєву двосторонню передачу мультимедіа через браузер без плагінів. WebRTC забезпечує скремблювання всіх медіапотоків (обов'язкове шифрування SRTP) за стандартом DTLS, тому вважається безпечним механізмом зв'язку.

Нижче наведено ключові характеристики цих протоколів за критеріями:

- Затримка: RTP завдяки UDP-базі і простій архітектурі забезпечує дуже низьку затримку передачі (загалом навіть мілісекунди), бо не використовує повторну доставку пакетів. WebRTC дає надзвичайно низьку затримку – звичайно суб-500 мілісекунд (0,5 с) від «тканини» камери до виводу на пристрій споживача. Однак у великомасштабних сценаріях із ретрансляцією через сервери час затримки може значно зростати (до десятків секунд) через необхідність перетворення потоків.
- Втрати пакетів: Оскільки RTP побудований поверх UDP, він не гарантує доставку – втрачені пакети просто ігноруються, що прийнятно для багатьох реалійнозонових застосувань. WebRTC також передбачає UDP-основу, але містить механізми компенсації втрат (наприклад, FEC чи опитування отримувача) для покращення якості при ненадійному з'єднанні.
- Адаптивність: RTP сам по собі не адаптивний до змін мережевих умов – джерело може передавати фіксовані потоки. Натомість WebRTC включає адаптивні алгоритми бітрейту (за потреби змінює якість потоку відповідно до пропускної спроможності каналу), але це призводить до деградації якості при поганому з'єднанні самого найслабшого учасника.
- Масштабованість: RTP традиційно використовується у сценаріях «точка-точка» або з малою кількістю одержувачів (наприклад, конференції SIP/RTP або багатоточковий multicast), оскільки сам по собі не призначений для CDH. WebRTC реалізований як peer-to-peer технологія і за замовчуванням не масштабується через CDN. Для великої аудиторії застосовуються серверні медіа-сервери (Selective Forwarding Unit) або перетворення в інші формати (див. нижче).

- Якість передачі: Обидва протоколи підтримують передачу високоякісного аудіо/відео (кодеки H.264, VP8/VP9/AV1 тощо). RTP мінімально «обтяжений» накладними витратами, тому потенційно може краще зберігати якість при надійних з’єднаннях. WebRTC гарантує високу якість завдяки адаптивним потокам, але в односторонній трансляції максимальна якість знижується до рівня найслабшого з’єднання (найменша пропускна спроможність серед глядачів).
- Безпека: RTP за замовчуванням не шифрується (потребує SRTP для захисту). WebRTC обов’язково шифрує медіапотоки за допомогою DTLS-SRTP; ненашифровані WebRTC-сесії заборонені стандартом IETF. Таким чином, WebRTC безпечніший «з коробки» порівняно з RTP.
- Сумісність: RTP є універсальним стандартом у мережевому стрімінгу, використовується у SIP-телефонії, відеоконференціях та вбудований у багато систем (наприклад, Skype, різні VoIP-сервери). WebRTC підтримується сучасними браузерами (Chrome, Firefox, Safari тощо) та використовується у веб-додатках (Jitsi Meet, BigBlueButton, Slack, Amazon Chime). Однак WebRTC вимагає специфічного стеку (STUN/TURN, peer-конект) і не зворотно сумісний із «чистим» RTP без додаткових перетворювачів.
- Ресурсоємність: RTP «легкий» на обробку – мінімальна накладна інформація, тому використовується навіть на вбудованих пристроях. WebRTC потребує додаткових ресурсів для шифрування, роботи з ICE та адаптації, але переважно виконується на клієнті. Масштабування WebRTC потребує значних обчислювальних ресурсів на сервері (SFU або MCUs), щоб обробляти багато потоків.

Приклади застосувань: RTP лежить в основі більшості VoIP-систем і мультимедійних сервісів реального часу (наприклад, Skype, Zoom, Webex при передачі медіа). WebRTC напряму використовується у браузерних вебконференціях та чатах (Jitsi Meet, BigBlueButton), а також у багатьох комерційних застосунках (Slack, Amazon Chime).

Сильні та слабкі сторони: RTP забезпечує найнижчу затримку і простоту реалізації, але не гарантує доставку пакетів і не шифрує медіа. Його сильна сторона – висока швидкість та універсальність (підтримка різними пристроями та програмами). Слабкі сторони RTP – відсутність захисту даних і обмежені засоби адаптації якості. WebRTC дає надзвичайно низьку затримку і гарантує безпеку (обов’язкове шифрування), підтримку сучасних кодеків і NAT-тунелювання, але складний у розгортанні і не масштабований «із коробки» на велике число споживачів. Також WebRTC потребує суттєвих ресурсів для обробки медіа та сигналізації.

Таблиця 3.2 – Порівняння протоколів реального часу RTP і WebRTC за критеріями ефективності

Критерій	RTP	WebRTC
Затримка	Дуже низька (пакетна, ~мс)	Наднизька (типово <0,5 с)
Втрати	Відсутність гарантій доставки (часткові втрати допустимі)	Підтримка ARQ/FEC, але усе одно допускає деякі втрати
Адаптивність	Без адаптивності (фіксований бітрейт)	Динамічна адаптація якості під мережу (алгоритми ABR)
Масштабованість	Обмежена (більші мережі через RTP не будуються без сервера)	Складна (P2P-архітектура, потрібні сервери для масштабних трансляцій)
Якість передачі	Висока за стабільного зв’язку, немає додаткових накладних	Висока (адаптація по мережі), але може падати до рівня найгіршого каналу

Продовження таблиці 3.2

Безпека	Не шифрується; можливе використання SRTP	Обов’язкове шифрування (SRTP); захищена сигналізація
Сумісність	Широко підтримується (VoIP, RTP-RTSP системи)	Підтримується браузерами (Chrome, Firefox, Safari) та вебдодатками
Ресурсоємність	Низька (простий UDP/RTCP)	Вища (шифрування, ICE, трансляція)

Протоколи RTP і WebRTC розраховані на мінімальну затримку і використовуються у реальному часі. RTP показує себе надійним у стійких мережах та простим у реалізації, але не має вбудованої безпеки і адаптивності. WebRTC гарантує безпеку і дуже низьку затримку від «дзвінка» до «відтворення», тому підходить для інтерактивного зв’язку. Однак його слабкість – обмежена масштабованість та залежність від якості мережі кінцевих користувачів, що ускладнює використання для масових трансляцій

2. Протоколи потокового мовлення: RTMP і SRT

RTMP (Real-Time Messaging Protocol) – застарілий протокол, спочатку розроблений Macromedia/Adobe для передачі потокового аудіо, відео і даних з Flash-плеєра на сервер. RTMP використовує TCP для встановлення стійкого з’єднання, що забезпечує гарантовану доставку, але з підвищеною затримкою. Протокол не підтримує найновіші кодеки (H.265, AV1) за замовчуванням і не містить вбудованого шифрування, хоча використовуються його розширення RTMPS (TCP+TLS). Незважаючи на недоліки, RTMP довго залишався стандартом інгесту у стрімінгу через широку підтримку з боку CDN та апаратних рішень. Наприклад, сервіси YouTube Live та Twitch приймають відеопотік від енкодерів через RTMP (або RTMPS) як вхідний медіапотік. Перевагою RTMP є стабільна передача без втрат (через TCP), а недоліком –

відносно великі затримки у декілька секунд та потреба у спеціальних серверах (та відсутність сумісності з HTTP/2).

SRT (Secure Reliable Transport) – сучасний відкритий протокол для потокової передачі відео, розроблений компаніями Haivision та Wowza. SRT базується на UDP, але реалізує власні механізми контролю помилок і повторної доставки (ARQ) та FEC (корекція втрат), забезпечуючи надійну доставку при збереженні низької затримки. Протокол підтримує шифрування AES (128/256 біт), що гарантує захищеність контенту. Завдяки цим властивостям SRT добре працює у «грязній» мережі Інтернет: мінімальна затримка може бути близько 0,12 с при використанні малих буферів, водночас відео передається без втрат. Головним недоліком є поки що обмежена підтримка серед апаратних пристроїв і програм: SRT тільки набирає популярності, тому для його підтримки потрібно спеціальне ПЗ або обладнання. Приклад використання SRT – стримінг-платформи та студії, що передають відео із студії у CDN через інтернет, а також нові енкодери (наприклад, пристрої Haivision, Matrox, деякі NAS) підтримують SRT.

Характеристики протоколів RTMP і SRT за критеріями:

- Затримка: RTMP через TCP забирає більше часу на встановлення і передачу (звичайно від кількох до десятків секунд в залежності від налаштувань і використання CDN). SRT за рахунок UDP-бази і малих буферів може забезпечувати затримку від десятків до сотень мілісекунд; при мінімальних буферах «скло» між записом і виводом може бути близько 120 мс.
- Втрати пакетів: RTMP гарантує відсутність втрат завдяки TCP (усі пакети перепередаються при пропаданні); SRT допускає втрати як наслідок обмежень бітрейту і буферів, але прагне корегувати їх за допомогою механізмів повторної доставки і FEC, тому фактично споживач бачить практично повну передачу без заметених кадрів.
- Адаптивність: RTMP не має адаптивного бітрейту: потік фіксований. SRT можна налаштувати з певним буфером на затримку, але він

не є адаптивним «з коробки» (потрібно регулювати бітрейт вручну). Проте в умовах мережевої нестабільності SRT автоматично коригує повторну доставку пакетів, забезпечуючи стабільність якості потоку.

- Масштабованість: RTMP традиційно призначений для прямого з'єднання енкодера з CDN або сервером трансляції, добре працює на великих масштабах через CDN (через політику push вхідних потоків). SRT, будучи відносно новим, підтримується переважно у точково-точкових зв'язках; CDN-провайдери ще починають додавати підтримку SRT. У разі великого попиту часто комбінують SRT для надходження на сервіс і потім трансформують у популярніші формати або поширюють через CDN.
- Якість передачі: RTMP забезпечує стабільну якість (H.264 тощо) без втрат кадрів (за умови стабільного каналу), але підвищена затримка може погіршувати реактивність. SRT орієнтований на «високоякісне відео», використовуючи всі переваги UDP (мінімальна затримка) та можливість одночасного шифрування (AES).
- Безпека: Зовні RTMP без захисту (крім RTMPS), тому не підходить для конфіденційного контенту без додаткових методів шифрування. SRT аплікаційно підтримує AES-шифрування, що робить його безпечним для передачі платного або комерційного контенту.
- Сумісність: RTMP все ще підтримується багатьма серверами і програмами (Adobe Media Server, Wowza, CDN, OBS тощо), хоча Flash-плеєри майже відпали. SRT поширюється через CRT-альянс і підтримується сучасними енкодерами і деякими CDN (наприклад, Zixi/LiveU мають SRT-опції). Однак SRT не сумісний з класичними Flash/RTMP-плеєрами.
- Ресурсоємність: RTMP через TCP більше навантажує CPU і мережу (підтримка постійного з'єднання, підтверджень). SRT використовує додаткові ресурси на реалізацію корекції помилок і шифрування, але загалом це не критично для сучасного апаратного забезпечення – мережеві енкодери оснащені для цього «з коробки».

Приклади застосування: RTMP широко використовувався у стрімінгу та до сьогодні залишається вживаним як протокол прийому входу (інгест) у системах прямої трансляції, наприклад YouTube Live, Twitch, OBS Studio – більшість цих систем приймають відеопотік від стрімера саме через RTMP (RTMPS) або обробляють його в межах CDN. SRT набуває популярності в корпоративному та професійному сегментах: наприклад, платформи LiveU, QNAP і деякі студії використовують SRT для передачі відео з місця події у студію або CDN із високою надійністю та низькою затримкою. Користь SRT особливо помітна при стрімінгу 4K/HD за ненадійного Інтернет-каналу (застосовується й у телеіндустрії).

Сильні та слабкі сторони: RTMP є перевіреним протоколом із широкою інфраструктурною підтримкою; його сильна сторона – проста інтеграція зі старими системами та гарантована доставка через TCP. Слабкі сторони – велика затримка, застарілість (немає сучасних кодеків без доопрацювання) і ненадійна безпека. SRT забезпечує «найкраще з обох світів» між UDP і TCP: високу надійність, низьку затримку, AES-шифрування; сильністю є адаптація до втрат і захист контенту. Слабкі сторони SRT – обмежена підтримка на боці деяких провідних CDN і необхідність налаштувань буфера. Через це SRT частіше застосовується там, де безпека і якість критичні, а не для звичайних Web-трансляцій.

Таблиця 3.4 – Порівняння протоколів потокового мовлення RTMP і SRT

Критерій	RTMP	SRT
Затримка	Помірна–висока (TCP: ~1–10 с)	Дуже низька (залежить від буфера, від ~0.12 с)
Втрати	Практично відсутні (TCP перепередає)	Мінімізовані (автоматичні ARQ/FEC)

Продовження таблиці 3.4

Адаптивність	Немає (фіксований бітрейт)	Динамічна корекція (ARQ), фіксований бітрейт
Масштабованість	Широко використовується з CDN (інгест)	Обмежена (зазвичай P2P; підтримка CDN зростає)
Якість передачі	Висока (втрат немає)	Висока (надійна передача навіть через лосі мережі)
Безпека	Немає (лише RTMPS через TLS)	Є (AES-128/256 шифрування)
Сумісність	Висока (сервери, пристрої, ПО)	Зростаюча (підтримка сучасних енкодерів і ПО)
Ресурсоємність	Помірна (TCP-стек, стабільне з'єднання)	Вища (реалізація FEC/ARQ і шифрування)

RTMP і SRT призначені для потокової передачі контенту з високою якістю, проте з різними пріоритетами. RTMP добре працює при стабільному зв'язку і в CDN-інфраструктурі, але страждає від значної затримки і відсутності захисту. SRT, навпаки, забезпечує низьку затримку та надійну доставку в непередбачуваних мережах завдяки власній корекції помилок. Таким чином, RTMP доцільний у традиційних стрімінгових рішеннях (кінцева якість важливіша за затримку), а SRT — у випадках, коли важливо поєднати низьку затримку з надійністю і безпекою (професійний IPTV, новинарні стріми, віддалені виробничі підрозділи)

3. Адаптивні HTTP-протоколи: HLS та DASH

HTTP Live Streaming (HLS) – адаптивний протокол потокового відео, розроблений Apple. У HLS відео розбивається на невеликі сегменти (звичайно 2–10 с), які зберігаються у форматі MPEG-TS або fMP4, а манифест .m3u8 вказує клієнтам послідовність сегментів. Клієнт динамічно вибирає сегенти з відповідним бітрейтом залежно від пропускної спроможності каналу. HLS широко підтримується на пристроях Apple та через JavaScript-бібліотеки на інших платформах.

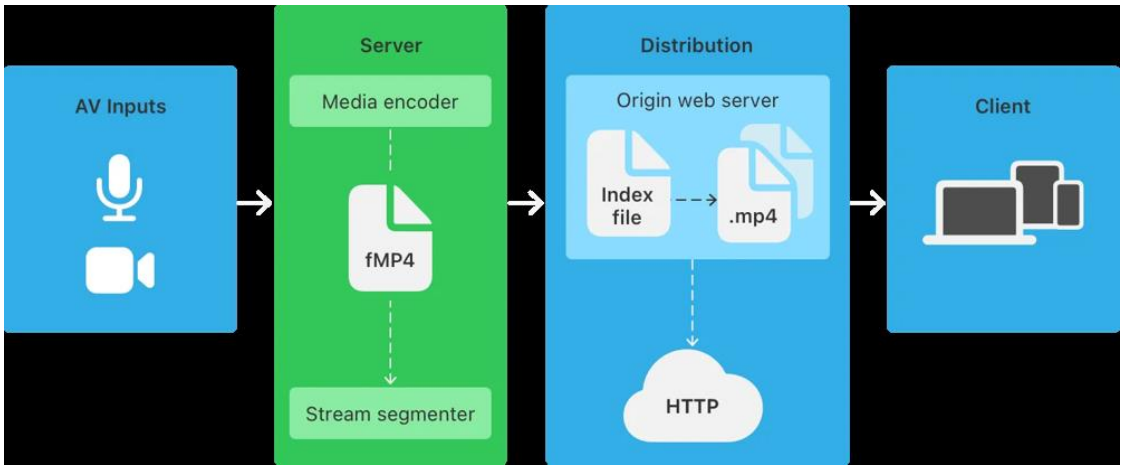

Рисунок 3.1 – Схематичне представлення принципу роботи адаптивнихHTTP-протоколів HLS та MPEG-DASH

DASH (Dynamic Adaptive Streaming over HTTP) – адаптивний протокол, ініційований MPEG. Він також розбиває контент на короткі сегменти, але використовує XML-маніфест MPD та контейнер ISO BMFF (CMAF). DASH не прив'язаний до конкретних платформ і дозволяє використовувати будь-які кодеки (H.264, VP9, AV1 тощо). На рівні концепцій обидва протоколи подібні (адаптивна потокова передача через HTTP/CDN), але відрізняються деталями реалізації та підтримкою.

Характеристики HLS і DASH:

- Затримка: Стандартний HLS показує високу затримку: через буфери і велику довжину сегмента (6 с) потрібно передіспівати щонайменше 3 сегменти (мінімум ~18 с затримки). Типові значення «скло-до-скло» складають десятки секунд (близько 15–30 с). Однак з виходом Low-Latency HLS (LL-HLS, IETF ed.2, 2020) затримку можна зменшити до ~5–10 с. DASH у стандартній версії дає меншу базову затримку (завдяки підтримці chunked transfer, SMOOTH та загалом меншим буферам) – зазвичай декілька секунд на ланцюжок сегментів. З LL-DASH і CMAF цей показник може знижуватись до кількох секунд.

- Втрати пакетів: HLS та DASH базуються на HTTP/TCP, тому гарантують доставку всіх сегментів (усі втрати мережі компенсуються TCP). Це підвищує надійність і консистентність якості відео (не втрачається ні сегмента), але уповільнює передачу при повторних запитах (тобто збільшує буферизацію).
- Адаптивність: Обидва протоколи реалізують ABR (адаптивний бітрейт): створюють кілька версій відео під різні швидкості, а клієнт обирає оптимальний варіант у реальному часі. HLS обмежений за вибором кодеків (специфікація Apple підтримує переважно H.264/HEVC для відео і AAC/AC3 для аудіо), а DASH – кодування є вільнішим (VP9, AV1 доступні). Менша різноманітність кодеків у HLS спрощує відтворення на пристроях Apple, але звужує гнучкість вибору формату.
- Масштабованість: HLS і DASH використовують стандартні HTTP-сервери та CDN, що робить їх надзвичайно масштабованими для великої кількості користувачів. Кешування сегментів і манифестів CDNами дозволяє обслуговувати мільйони глядачів без додаткового навантаження на сервер.
- Якість передачі: Обидва протоколи можуть передавати відео високої роздільної здатності (HD/4K). Через ABR вони завжди намагаються підтримувати максимальну якість для поточного каналу; якість дещо вища в тому варіанті, який клієнт обирає. Оскільки передача надійна (всі сегменти доставляються), результатом є відтворення без пропусків кадрів, хоч і з буферними затримками.
- Безпека: HLS підтримує вбудоване шифрування сегментів (AES-128 зі спільним ключем, також інтеграцію з Apple FairPlay DRM). DASH використовує стандарт Common Encryption (CENC) і може працювати з кількома системами DRM (Widevine, PlayReady тощо). В обох випадках контент може бути надійно захищений, проте реалізація DRM-систем часто складна і залежить від платформи.
- Сумісність: HLS — індустріальний стандарт від Apple, нативно підтримується на iOS/MacOS (Safari, AVPlayer) і більшістю Android-додатків

(ExoPlayer), а також через MSE-бібліотеки в інших браузерах. DASH не підтримується в браузерах Apple без додаткових скриптів – для iOS/Safari потрібні сторонні плеєри; але він легко працює на Android і в десктопних браузерах через MSE. У цілому HLS працюватиме практично скрізь, DASH – скрізь, окрім «яблучних» пристроїв без додаткових компонентів.

- Ресурсоємність: Оскільки обидва протоколи базуються на HTTP, вони використовують потенціал кешування та CDN, знижуючи навантаження. Генерація сегментів може підвищувати навантаження на сервер-відправник (енкодер має нарізати відео), але це поза рамками протоколу. Від клієнта вимагається досить пам’яті для буфера і декодування, але сучасні пристрої з цим впораються. У порівнянні з несегментованими потоками (RTMP, WebRTC), ABR-протоколи потребують більше мережевих запитів (HTTP-GET на кожний сегмент), але це компенсується значною масштабованістю.

Приклади застосування: HLS є стандартом для широкомасштабних стрімінгових сервісів та VoD. Наприклад, Apple TV+, Netflix та Disney+ можуть використовувати HLS для деяких платформ; також багато телевізійних сервісів (Hulu Live, ESPN+) вдаються до HLS. DASH найчастіше застосовується у сервісах OTT (Netflix, YouTube, Amazon Prime Video, Hulu), особливо там, де потрібна відкрита платформа і підтримка різних кодеків. Наприклад, Netflix традиційно використовує MPEG-DASH з DRM.

Порівняльні зауваги: HLS характеризується широкою підтримкою і надзвичайною масштабованістю через HTTP/CDN, але має відносно високу мінімальну затримку (18–20 с за стандартної конфігурації). Основний недолік HLS – вузький набір підтримуваних кодеків (переважно H.264/HEVC) і прив’язка до Apple DRM (FairPlay). DASH, навпаки, є відкритим стандартом без ліцензійних обмежень, підтримує більш широкий спектр кодеків (VP9, AV1 тощо) і історично забезпечував нижчу затримку. Слабкі сторони DASH – відсутність нативної підтримки на пристроях Apple і фрагментованість екосистеми (різні DRM, різні плеєри). Зокрема, розділення на різні реалізації робить налагодження потоків складнішим.

Таблиця 3.5 – Порівняння адаптивних HTTP-протоколів: HLS та DASH

Критерій	HLS	DASH
Затримка	Висока (стандартно ~15–30 с); з LL-HLS ~5–10 с	Середня–низька (залежить від сегментації; зазвичай ~3–10 с; з LL-DASH < 5 с)
Втрати	Відсутні (HTTP/TCP), високоповторність	Відсутні (HTTP/TCP)
Адаптивність	Повна (ABR, від 144p до 4K; підтримка субтитрів, аудіодоріжок)	Повна (ABR, широке коло кодеків, підтримка обміну ABR)
Масштабованість	Дуже висока (CDN, кешування)	Дуже висока (CDN, кешування)
Якість передачі	Висока (адаптація, підтримка 4K/HD)	Висока (кодек-агностик, 4K/HD)
Безпека	AES-шифрування сегментів, DRM FairPlay	CENC + різні DRM (Widevine, PlayReady тощо)
Сумісність	Нативно на Apple; на інших – через MSE/плеєри	Мультіплатформенний; потребує MSE або додатків на iOS
Ресурсоємність	Висока для серверу (нарізка сегментів); клієнт – помірна (буфер)	Аналогічна HLS (HTTP/TCP)

HLS і DASH – це два основні протоколи адаптивного стрімінгу. HLS домінує в екосистемі Apple і гарантує високу масштабованість через CDN, але традиційно потерпає від великої затримки (створює буфер понад 15 с). DASH навпаки забезпечує ширшу гнучкість (відсутність «прив’язки» до конкретних кодеків або DRM) і потенційно нижчу базову затримку при подібному налаштуванні. Для live-стрімів часом вибір між ними визначає сумісність: HLS потрібен для iOS/Safari, тоді як DASH підходить для Android/Web. Low-latency версії (LL-HLS/LL-DASH) дозволяють наблизити обидва протоколи до реального часу (~5–10 с) з невеликою втратою надійності. Загалом,

HLS/DASH є оптимальними для великих аудиторій з потрібністю масштабування і ABR, але малопридатні для інтерактивних застосувань (де краще WebRTC).

4. Альтернативні транспортні протоколи: QUIC, DCCP, UDP-Lite

QUIC (Quick UDP Internet Connections) – транспортний протокол нового покоління, розроблений Google та стандартизований IETF. QUIC працює поверх UDP і інтегрує в собі можливості TCP (надійність передачі, керування потоком і перевантаженням) та TLS 1.3 для безпеки. Основні особливості QUIC: швидке встановлення з'єднання (з TLS-шифруванням за один хендшейк), мультиплексування кількох потоків без блокування по Head-of-Line, підтримка переміщення з'єднання між мережами (корисно для мобільних). За рахунок цього QUIC зменшує затримку у порівнянні з TCP приблизно на 30–33% (швидше встановлення з'єднання і менший TTFB). QUIC обов'язково шифрує весь трафік (TLS 1.3 на транспортному рівні), що робить його набагато безпечнішим за відкритий UDP. Цей протокол лягає в основу HTTP/3, тому вже використовується Google (YouTube, Gmail), Facebook, Cloudflare та іншими великими сервісами для доставки даних з поліпшеною продуктивністю. QUIC добре справляється в умовах ненадійних мереж (приблизно на 55% краща робота при втраті, ніж TCP), а також забезпечує плавне перемикання мереж (без розриву потоків) при змінах зв'язку.

DCCP (Datagram Congestion Control Protocol) – транспортний протокол, створений як «невідмовоздатний» аналог TCP. DCCP забезпечує управління перевантаженням (як TCP), але не гарантує доставку всіх даних (базується на повідомленнях/датаграмах). DCCP забезпечує встановлення з'єднання і підтвердження (аналогічно TCP), проте старі пакети можуть бути відкинуті без повторної доставки, якщо вони втратились або застаріли. Така поведінка корисна для мультимедіа у реальному часі: у випадку зникнення пакета програвач «телевізійно» спотворює відео, замість затримки на очікування. DCCP передбачає механізми вибору алгоритму керування перевантаженням

(CCID) та використовує Explicit Congestion Notification (ECN). Даний протокол пропонує баланс між швидкістю UDP та порядком TCP. Використовується переважно в мережевих іграх, VoIP, потоковому аудіо/відео (додатки, де старі пакети швидко «старіють» і не потребують доставки). Зазвичай DCCP не несе шифрування і має обмежену підтримку у стандартних ОС (але реалізований у Linux/FreeBSD).

UDP-Lite (Lightweight UDP) – варіант UDP з полегшеним контрольним підсумком (checksum). На відміну від звичайного UDP, де або весь пакет перевіряється, або він відкидається, UDP-Lite дозволяє частково верифікувати тільки початок пакета, доставляючи пакет із можливими пошкодженнями, якщо вони не критичні. Це корисно для деяких мультимедіа-кодеків, де незначні помилки в кадрі менш помітні, ніж пропуски кадрів. Наприклад, деякі аудіокодеки (AMR) та технології IP-телефонії використовували UDP-Lite для покращення якості при втраті даних. UDP-Lite не містить механізмів управління затримкою чи перевантаженням – він так само легкий, як UDP, але з можливістю допустити биті байти. Підтримується у деяких ядрах ОС (Linux, FreeBSD). Головні переваги – мінімальні затримки (як у UDP) і можливість «пропустити» пошкодження через застосунок. Недоліки – дуже мала поширеність у стандартних мережевих стеків і відсутність додаткових сервісів (немає управління потоком, не шифрується).

Порівняльні характеристики за критеріями:

- Затримка: QUIC завдяки швидкому хендшейку і мультиплексуванню знижує час до першого байта і загальну затримку у порівнянні з TCP. DCCP близьке до UDP – додатковий трюк з хендшейком, тому затримка нижча, ніж у TCP, але вища, ніж у чистого UDP через CC機. UDP-Lite має ту ж низьку затримку, що й UDP (мінімальні накладні).
- Втрати пакетів: QUIC, як і TCP, гарантує доставку, але краще поводиться при паралельних потоках (немає блокування). DCCP не гарантує доставку (пакети зникають при перевантаженні), зате повідомляє про втрату (і

використовує ECN). UDP-Lite також не гарантує доставку всіх даних (як UDP), але не кидає пакети через биті кадри – програма отримує те, що є.

- Адаптивність: QUIC включає сучасні алгоритми congestion control (Cubict, BBR) і дозволяє інтеграцію з HTTP/3 на рівні адаптивних алгоритмів. DCCP дозволяє вибір алгоритму CCID, проте сам по собі не адаптивний «підтримує управління перевантаженням, але не бітрейт»; це залишає адаптацію програмі. UDP-Lite не має жодних власних CC, оскільки рахується просто варіантом UDP.
- Масштабованість: QUIC, використовуючись у HTTP/3, масштабований інтернет-серверами (Google, Cloudflare) і CDN. DCCP та UDP-Lite навряд чи використовуються в CDN-архітектурах – це більш нішеві протоколи. DCCP може масштабуватися у мережах, що підтримують цей протокол, але сумісність обмежена. UDP-Lite не підтримується багатьма пристроями, тому на практиці мало застосовується.
- Якість передачі: За відсутності втрат, кожен з них може передавати будь-яку інформацію (різниця в упаковці). UDP-Lite потенційно покращує «деґрадування» відео/аудіо при битих пакетах: у разі помилок битий кадр частково декодується, а не викидається повністю. DCCP, наприклад, у VoIP дозволяє «засвистіти» звук замість повної тиші (деякі дані доходять). QUIC (через TCP-подібну надійність) передає контент без втрат – більше підходить для будь-яких медіа (і, зокрема, відновлення втрачених пакетів, як у звичайному HTTP).
- Безпека: QUIC завжди шифрує трафік (TLS 1.3 на транспортному рівні). DCCP і UDP-Lite не передбачають вбудованого шифрування або автентифікації – їм потрібні додаткові рівні (DTLS або IPSec за окремими механізмами). UDP-Lite не несе безпеки за замовчуванням.
- Сумісність: QUIC поширюється дуже широко (застосовується у всьому інтернеті через HTTP/3). DCCP і UDP-Lite підтримуються лише в деяких ОС або обладнанні. Наприклад, UNIX-збудаваки мають опції DCCP, а UDP-Lite був доданий у ядро Linux і FreeBSD. В Windows ці протоколи

практично не застосовуються. Сторонні мережеві бібліотеки часто не підтримують їх «з коробки».

- Ресурсоємність: QUIC має накладні витрати на шифрування і складну логіку (хоча значну частину роботи можуть брати на себе апаратні TLS-ускорювачі). DCCP додає мало додаткового навантаження (це UDP+заголовок CC), проте вимагає обробки Congestion Control; UDP-Lite практично не відрізняється від UDP (ті ж накладні, що й UDP).

Таблиця 3.6 – Порівняння альтернативних транспортних протоколів: QUIC, DCCP, UDP-Lite

Критерій	QUIC	DCCP	UDP-Lite
Затримка	Дуже низька (UDP+інтегрований TLS handshake, ~0-RTT)	Низька (UDP+заголовок DCCP; дещо більше за UDP через хендшейк)	Низька (така ж, як у UDP)
Втрати	Гарантована доставка (як у TCP, але без head-of-line)	Часткова (без гарантованої доставки; старі пакети ігноруються)	Без гарантії (пакети можуть прийти пошкодженими)
Адаптивність	Вбудовані сучасні алгоритми congestion control, підтримка HTTP/3	Можливість вибору CCID; власних адаптивних механізмів немає	Немає (як UDP)
Масштабованість	Висока (HTTP/3 у всьому інтернеті)	Обмежена (має невелику підтримку на рівні ОС)	Обмежена (незначна підтримка)

Продовження таблиці 3.6

Якість передачі	Дуже висока (шифрований і надійний потік без втрат)	Висока для реального часу (надає послідовний потік із пропусками)	Для мультимедіа – потік з допусками битих кадрів
Безпека	Обов’язкове шифрування (TLS 1.3)	Немає вбудованого (потрібно DTLS, IPSec)	Немає вбудованого
Сумісність	Широко впроваджений (браузери, сервери, CDN)	Обмежена (підтримка в Linux/FreeBSD)	Дуже обмежена (Linux/FreeBSD)
Ресурсоємність	Помірна (шифрування TLS, мультиплексування)	Низька (легкий протокол, додаткові поля)	Дуже низька (мінімум накладних, частковий checksum)

QUIC, DCCP і UDP-Lite – це альтернативи традиційним протоколам (TCP/UDP) із специфічними цілями. QUIC є фактично майбутнім веб-транспорту: він поєднує надійність TCP з перевагами UDP, зменшує затримку підключень і забезпечує безпеку. DCCP – це гібрид TCP/UDP із вбудованим контролем перевантаження, що підходить для застосунків, які потребують керування мережею, але можуть терпіти втрати (ігри, VoIP). UDP-Lite, навпаки, жодного порядку чи контролю не додає, зате доповнює UDP можливістю допустити помилкові біти – це краще підходить для мультимедіа на мобільних каналах, де часткові пошкодження прийнятні. Застосування цих протоколів обмежене: QUIC стрімко поширюється в інтернеті як основа HTTP/3, тоді як DCCP і UDP-Lite більше вивчаються як дослідницькі або нішеві рішення. Загалом QUIC поєднує малу затримку і надійність (і вже застосовується у великих сервісах), DCCP дає часткову надійність із контролем перевантаження для стрімінгу/ігрових застосунків, а UDP-Lite спеціалізовано для підвищення якості мультимедіа за поганої мережі

3.5 Висновки з розділу 3

У цьому розділі було проведено порівняльний аналіз основних протоколів передачі даних, які використовуються в електронних комунікаційних мережах для мультимедійного мовлення прямих трансляцій. Розглянуто як традиційні транспортні протоколи TCP та UDP, так і спеціалізовані мультимедійні протоколи реального часу (RTP, WebRTC) та потокового мовлення (RTMP, SRT, HLS, DASH).

Аналіз показав, що:

- Протокол TCP забезпечує високу надійність і контроль доставки, однак створює значні затримки, що робить його менш придатним для реального часу;
- UDP, навпаки, демонструє мінімальні затримки, але не гарантує доставку, що обмежує його застосування без додаткових протоколів верхнього рівня;
- Протоколи RTP і WebRTC більш адаптовані до задач реального часу, проте WebRTC має вищу інтеграцію з браузерами та сучасними системами відеоконференцій;
- Протоколи RTMP і SRT мають різні переваги залежно від контексту: RTMP — у стабільних локальних мережах, SRT — у мережах з нестабільною пропускною здатністю;
- HLS та DASH є найкращими для масштабованого розповсюдження відео на вимогу завдяки адаптивній зміні бітрейту відповідно до умов мережі.

Загалом, вибір протоколу значною мірою залежить від конкретного сценарію використання — затримки, вимог до якості, надійності доставки, сумісності з клієнтськими пристроями та доступних ресурсів. Отримані результати дозволяють сформувати обґрунтовані рекомендації щодо вибору протоколів для побудови ефективних мультимедійних систем прямого мовлення.

4 ПЕРСПЕКТИВИ РОЗВИТКУ ТА ПРАКТИЧНІ РЕКОМЕНДАЦІЇ

4.1 Оптимізація вибору протоколу залежно від сценарію трансляції

У сучасному відеостримінгу застосовується велика кількість мережевих протоколів, кожен із яких має свої особливості та призначення. Оптимальний вибір залежить від конкретного сценарію трансляції, технічних обмежень, вимог до затримки, надійності, адаптивності, масштабованості й безпеки.

Для сценаріїв відео за запитом (VoD) або масових трансляцій найчастіше використовують HTTP-базовані протоколи HLS та MPEG-DASH. Їх перевага — простота масштабування через CDN і підтримка адаптивної передачі бітрейту (ABR). Вони гарантують надійність доставки за рахунок TCP, проте мають значну затримку (6–30 секунд). У випадках, коли латентність не є критичною, такі протоколи забезпечують стабільне відтворення на широкому спектрі пристроїв. Крім того, вони добре інтегруються з браузерами та мобільними платформами, а також підтримуються у фреймворках, які автоматично перемикають якість потоку відповідно до умов мережі користувача.

Протокол RTMP, хоч і втрачає актуальність, залишається поширеним для інжесту відео — передачі потоку від енкодера до сервера. Подальша доставка до глядачів зазвичай здійснюється через HLS або DASH. RTMP має відносно низьку затримку, але обмежену підтримку у сучасних браузерах, особливо після відмови від Flash. Незважаючи на це, RTMP часто обирається через його просту реалізацію в програмному забезпеченні та сумісність з багатьма потоковими платформами.

Коли головним пріоритетом є мінімальна затримка — як у спортивних трансляціях, відеоконференціях або інтерактивних шоу — застосовуються UDP-орієнтовані протоколи, зокрема WebRTC та SRT. WebRTC забезпечує затримку менше секунди, підтримується безплагінно у браузерах і призначений для двосторонньої реального часу комунікації. Його використання зростає в додатках для онлайн-зустрічей, ігор та телемедицини.

Протокол SRT краще підходить для нестабільних мереж: він має вбудовану корекцію втрат і шифрування, що робить його надійним для професійного відеострімінгу. Водночас, SRT забезпечує можливість роботи через публічні мережі, зменшуючи потребу в дорогому інфраструктурному обладнанні.

Також до UDP-протоколів нового покоління належать RIST і QUIC. RIST розроблений для наднадійної доставки в складних мережах і використовує вдосконалені алгоритми обробки втрат, що дозволяє підтримувати якість відео навіть за умов високої варіативності з'єднання. QUIC, як основа HTTP/3, зменшує затримки встановлення з'єднань і підвищує продуктивність стрімінгу через мультипотоковість, а також пропонує вбудовану підтримку шифрування, що особливо важливо для комерційних трансляцій.

Ключові фактори, які впливають на вибір:

- Затримка: критична для живих і інтерактивних трансляцій. Найнижча у WebRTC, нижча серед UDP-протоколів у SRT та RIST.
- Надійність: HLS/DASH (TCP) гарантують доставку, а SRT і RIST компенсують втрати через ARQ.
- Адаптивність: реалізується у HLS/DASH, частково — у WebRTC, обмежено — в SRT.
- Масштабованість: максимальна у HTTP-протоколів завдяки CDN; UDP-протоколи потребують спеціалізованої інфраструктури.
- Безпека: найвищий рівень у WebRTC, QUIC та SRT, завдяки вбудованому шифруванню.

Платформа також має значення: браузери підтримують WebRTC та HTTP-стрімінг, мобільні ОС віддають перевагу HLS (iOS) або DASH (Android), CDN найкраще працюють із сегментованим HTTP-контентом. Крім того, важливо враховувати можливості енкодерів, які часто мають підтримку RTMP або SRT, а також сумісність із плеєрами та сервісами кінцевого споживача.

З урахуванням вимог до трансляції, можна сформувати базові рекомендації:

- VoD або масштабна трансляція: HLS або DASH.
- Жива трансляція з помірною затримкою: Low-Latency HLS або DASH.
- Інтерактивна або двостороння комунікація: WebRTC.
- Погані мережі або висока вірогідність втрат: SRT або RIST.

Таким чином, вибір протоколу повинен базуватися на балансі між затримкою, якістю, надійністю та можливостями інфраструктури. У правильно підібраному поєднанні протокол дозволяє досягти максимальної якості трансляції у заданих умовах. Враховуючи швидку еволюцію технологій та зростання очікувань користувачів, гнучкий підхід до вибору протоколу — ключ до ефективного та конкурентоспроможного стримінгу.

4.2 Вплив новітніх технологій (5G, edge computing, AI-адаптація)

Розвиток мереж п’ятого покоління (5G), технологій обчислень на краю мережі (edge computing) та інтелектуальної адаптації мережевого трафіку суттєво розширює вимоги до протоколів мультимедійної трансляції. 5G забезпечує надзвичайно високу пропускну здатність і вкрай низьку затримку, що дозволяє зменшувати буфери і оперативно передавати дані. Edge computing (MEC) наближає серверні ресурси до кінцевих користувачів, а AI-алгоритми оптимізують розподіл пропускної здатності та адаптивне кодування. Разом ці фактори знижують небезпеку джиттера і розширюють можливості масштабування сервісів, вимагаючи від протоколів вищої гнучкості та продуктивності.

Вплив мереж 5G на мультимедійну трансляцію

Мережі 5G відрізняються дуже коротким часом радіозатримки (іноді менше 10 мс) та вищою сумарною швидкістю (до декількох Гбіт/с на користувача). Це створює передумови для реалізації практично у реальному

часі відеотрансляцій та стрімінгу в ультра-високій чіткості. Низька латентність 5G дозволяє суттєво скоротити буферизацію в протоколах, розширюючи можливість використання WebRTC і SRT не лише для відеоконференцій, але й для масового стрімінгу. Підвищена пропускна здатність надає запас по швидкості передачі, що дозволяє підтримувати кілька відеопотоків високої роздільної здатності одночасно. Крім того, у 5G-контексті з'являються такі механізми, як мережеве шарювання (network slicing), яке дозволяє гарантувати відведені ресурси для мультимедіа-сервісів, і комбінація з MEC – розміщення CDN чи SFU-серверів безпосередньо в базовій станції. Це зменшує затримку передачі та сприяє стабілізації пропускної здатності.

Ключові зміни при впровадженні 5G впливають на адаптивність протоколів наступним чином:

- Зниження затримки: Протоколи з низькими буферами (наприклад, WebRTC, SRT) дозволяють використати переваги 5G, здійснюючи мінімальну буферизацію й оперативне оновлення кадрів. Унаслідок цього вимоги до швидкого відновлення втрачених пакетів залишаються критичними, проте затримку від їх відновлення стають нижчими.
- Підвищення пропускної здатності: Високошвидкісні канали з 5G дозволяють обмінюватися мультимедіа вищої якості. Протоколи адаптивної передачі (HLS/DASH) можуть використовувати більш агресивні профілі кодування та збільшувати кількість паралельних потоків (тематичні доріжки, 360°-відео тощо). У свою чергу, WebRTC та QUIC ефективніше мультиплексують дані, повною мірою використовуючи ресурс.
- Динамічне керування: Завдяки інтелектуальним механізмам маршрутизації й планування ресурсів у 5G мережах, протоколи трансляції можуть координуватися із мережею для підтримки якості (QoS). Наприклад, протокол WebRTC використовує RTCP для збору статистики і може швидко адаптувати бітрейт відповідно до сигналів від мережі 5G. Подібно, 5G

забезпечує більш передбачувані умови (менше варіацій затримки), тож алгоритми адаптивної потокової передачі (ABR) працюють ефективніше.

Обчислення на краю мережі (Edge Computing) та оптимізація QoS

Технології edge computing передбачають перенесення обчислювальних та кешуючих ресурсів ближче до користувача, наприклад, у мобільні базові станції. Це дозволяє обробляти відео (транс-кодер, мультиплексор), аналіз трафіку та прогнозування на локальному рівні. Унаслідок цього знижується затримка передачі (шлях даних менший), зменшується джиттер і навантаження на магістральні канали. Типові переваги edge-обчислень у мультимедіа-трансляції включають:

- Швидке кешування: Розміщення популярного контенту або сегментів відео на граничних серверах дозволяє клієнтам завантажувати дані з найближчого джерела. HLS/DASH стримери отримують сегменти із локального кеша, що значно прискорює старт та адаптацію потоку.
- Локальне оброблення: Для задач, чутливих до затримки (напр., доповнена реальність, геймінг), відео може бути попередньо оброблено на узлі edge-сети (зміна бітрейту, швидка компресія, композицін різних відеодоріжок). Це зменшує обсяг трафіку і стабілізує якість.
- Розподілене масштабування: За допомогою MEC можна динамічно розгортати інстанси SFU, MCU або CDN-сервера поряд із великими скупченнями користувачів. Таке масштабування «на вимогу» забезпечує високий QoS навіть при піках навантаження.

ШІ-адаптація QoS в мультимедійній трансляції

Використання штучного інтелекту (ШІ) у QoS-менеджменті дозволяє протоколам стати «розумнішими» щодо стану мережі та якості відео. Наприклад, нейромережі та методи машинного навчання можуть передбачати коливання пропускної здатності та реагувати на них завчасно. Завдяки цьому ABR-алгоритми (Adaptive Bitrate) в HLS/DASH-клієнтах переходять на оптимальний профіль відео до того, як відбудеться різкий спад швидкості.

ШІ також застосовується для аналізу користувацьких взаємодій і якості сприйняття (QoE) у реальному часі. Протоколи можуть використовувати ці дані для коригування параметрів: наприклад, зменшення роздільності або фреймрейту в активних сцепів при збереженні центральних об’єктів у найвищій якості. Крім того, алгоритми штучного інтелекту здатні оптимізувати мережеві ресурси при плануванні мультиканальних потоків, балансуючи навантаження між кількома шляхами (наприклад, 5G та Wi-Fi одночасно) або вибираючи кращі базові станції для трансляції.

У цілому ШІ підвищує адаптивність протоколів шляхом автоматичного навчання на історичних даних трафіку і передачі знань про мережеві умови в операційну логіку (наприклад, інтеграція AI-конгешн-контролю в QUIC). Це забезпечує більш плавне масштабування, менші сплески затримки і більш ефективну компресію відповідно до поточних обмежень мережі.

Адаптивність та продуктивність протоколів WebRTC, SRT, QUIC, HLS/DASH

З огляду на покращені характеристики мереж та наявність інтелектуального управління ресурсами, кожен з основних мультимедійних протоколів розвивається своїм шляхом:

- WebRTC: Проєктований для реального часу, він природно використовує мінімальні затримки. У 5G-мережах WebRTC клієнти можуть зменшувати глибину джиттер-буфера, оскільки мережа менш мінлива. Для масштабування відеоконференцій WebRTC зазвичай працює через SFU (Selective Forwarding Unit), яка просто ретранслює потоки без зміни, що суттєво знижує навантаження на сервер. SFU легко розміщувати на edge-серверах, що ще більше зменшує затримку. WebRTC також інтегрує SRTP для захищеної передачі, що в сучасному контексті 5G підтримує конфіденційність і пріоритетність даних.
- SRT (Secure Reliable Transport): Цей протокол спроєктований для стабільної доставки відео з низькою затримкою по ненадійних мережах. Він

виконує моніторинг мережі в реальному часі та використовує механізми ARQ для швидкого відновлення втрачених пакетів. У умовах 5G зі стабільнішим каналом SRT може працювати з найнижчими налаштуваннями затримки, забезпечуючи практично безперервну передачу. Оскільки SRT має вбудоване AES-шифрування, він відповідає сучасним вимогам безпеки. Крім того, SRT легко розгортати у розподілених мережах: наприклад, відеопотік з камери може передаватися через декілька SRT-серверів, розміщених на краю мережі, щоб зменшити затримку та уникнути перевантаження магістралі.

- QUIC (HTTP/3): Це транспортний протокол на основі UDP з вбудованим TLS-шифруванням та мультиплексуванням потоків. QUIC усуває проблему блокування заданнями (head-of-line blocking) характерну для TCP, що особливо актуально в умовах високої пропускної здатності 5G. На його базі розвиваються нові способи передачі мультимедіа (наприклад, WebTransport) – мультистрокове двонаправлене з'єднання над HTTP/3, що підходить для інтерактивних відео- та геймінгових додатків. QUIC також активно підтримує розширення під мережі 5G, наприклад, функції multipath, які дозволяють одночасно використовувати декілька з'єднань (5G+Wi-Fi) для підвищення надійності. Завдяки гнучкій схеми передавання даних QUIC-потоки можуть динамічно підлаштовуватись під зміну пропускної здатності і затримки.
- HLS/DASH (HTTP Adaptive Streaming): Ці протоколи спочатку були заточені під надійну доставку через HTTP-сервери та CDN. Щоб відповідати новим вимогам низької затримки, вони отримали спеціальні розширення. Зокрема, введено LL-HLS – розширення Apple HLS, яке дозволяє розбивати сегменти на «частинки» для швидшої публікації, і подібний механізм у DASH, заснований на фрагментованих MP4 (CMAF). Такі зміни дають змогу досягати затримки в декілька секунд між відеозаписом і відтворенням. Водночас загальна модель адаптивної передачі збереглася: клієнт HLS/DASH використовує ABR-алгоритм, що враховує прогнози пропускної здатності, часто на основі AI, щоб динамічно змінювати якість відеопотоку. У поєднанні з CDN та кешуванням на краю це дозволяє цим

протоколам масово масштабуватися, забезпечуючи високу стабільність і якість при змінних мережевих умовах.

Відповідь протоколів на сучасні виклики

Нові можливості мереж і обчислювальної інфраструктури супроводжуються власними викликами. Протоколи трансляції реагують на них так:

- Масштабування: HLS/DASH традиційно масштабується завдяки CDN та кешуванню контенту за географічним принципом. У сценаріях реального часу (відеоконференції, live-стріми) WebRTC використовує архітектуру SFU або MCU, щоб організувати багатокористувацьку трансляцію без створення стічних з'єднань на кожного клієнта. Використання SFU на edge-серверах дозволяє дуже швидко збільшувати кількість учасників із мінімальним додатковим навантаженням. QUIC і HTTP/3 дозволяють відкривати багато паралельних потоків (multistream), що прискорює відправку вмісту і полегшує горизонтальне масштабування для сервісів із великою кількістю одночасних користувачів.
- Джиттер: Щоб зменшити коливання затримки, протоколи застосовують буферизацію і корекцію часу відтворення. WebRTC підтримує адаптивні буфери та динамічну зміну частоти кадрів, що допомагає компенсувати мережевий джиттер. SRT має механізми синхронізації часу передавання (timestamping) і швидке повторне передавання втрачених пакетів, що знижує вплив випадкових затримок. У 5G-мережах із підтримкою MEC джиттера стає менше, але протоколи все одно залишаються готовими до раптових коливань, наприклад, при переході користувача між базовими станціями.
- Передбачення пропускної здатності: Протоколи ABR (Adaptive Bitrate) HLS/DASH постійно аналізують швидкість мережі та можуть використовувати моделі машинного навчання для прогнозу майбутньої пропускної здатності. Це дає змогу переключатися на відповідну якість до того, як виникне деградація. QUIC використовує сучасні алгоритми керування

заторами (конгешн-контроль) – наприклад, NewReno, BBR та їх модифікації – що оперативно реагують на зміну пропускної здатності каналів. У 5G обирається пріоритетний клас обслуговування, протоколи інтерпретують ці відомості для налаштування bітрейту або чергопрацювання.

- Стиснення: Щоб ефективно використовувати підвищену пропускну здатність 5G, впроваджуються нові кодеки (AV1, H.265/HEVC, VVC), але на рівні протоколів важливі також механізми зменшення накладних витрат. QUIC, наприклад, застосовує QPACK-алгоритм стиснення заголовків у HTTP/3. Прагнучи мінімізувати затримки, деякі системи допускають адаптивне перезаписування (ре-кодування) потоку на краю мережі – наприклад, жорсткіші налаштування стиснення під час навантаження, збереження ключових кадрів для точності реконструкції.
- Кешування на краю: Edge-інфраструктура передбачає локальне зберігання контенту. У випадку HLS/DASH це означає, що плейлісти і сегменти можуть доставлятися з найближчого edge-CDN. WebRTC/SFU та QUIC-додатки можуть кешувати часто використовувані фрагменти чи статичні дані (аудіо, зображення тощо) для прискорення встановлення нових з’єднань. Наявність кешованого контенту на краю зменшує затримку початку відтворення і забезпечує більшу стійкість до збоїв магістралі.

Модифікації та нові профілі протоколів

Сучасні тенденції породили спеціальні розширення і профілі протоколів:

- LL-HLS (Low-Latency HLS): Розширення HLS-протоколу для досягнення наднизької затримки. Воно розбиває звичайні сегменти на менші «частинки» (partial segments) і використовує додаткові теги плейлиста (#EXT-X-PART, #EXT-X-SERVER-CONTROL) для негайної публікації. Це дозволяє забезпечувати інтервал від захоплення кадру до його відтворення в клієнта лише в кілька секунд замість десятків.
- CMAF (Common Media Application Format): Загальний формат контейнера для HLS та DASH, що спрощує упаковку мультимедіа. Він

передбачає фрагментовані MP4/TS-контейнери зможливістю видачі «чункованих» сегментів. CMAF-формат уніфікує стандарти, дозволяючи використовувати одне кодування для різних протоколів, і знижує латентність за рахунок гнучкого керування фрагментацією.

- WebTransport: Новий API/протокол для браузерів на основі HTTP/3/QUIC. Він пропонує двонаправлену передачу даних із можливістю мультиплексування та керування на рівні додатка. WebTransport спеціально розроблений для інтерактивних потоків (VR, ігри, реального часу відео), де необхідна мінімальна затримка і великий масштаб зв’язків. Порівняно з WebSocket, він використовує усі переваги QUIC (TLS-шифрування, мультистрім, можливість використання UDP).
- SRTP (Secure Real-time Transport Protocol): Протокол безпечного RTP, який забезпечує шифрування й аутентифікацію медіа-потоків у реальному часі. У сучасних потокових системах з безпечними мережевими вимогами SRTP є стандартним для WebRTC та інших протоколів реального часу. Він гарантує захист контенту на передачі й відповідає суворим вимогам конфіденційності в умовах 5G.
- SFU (Selective Forwarding Unit): Хоча це не мережевий протокол, SFU – архітектурне рішення для масштабування відеоконференцій. SFU приймає потоки від клієнтів і пересилає їх іншим учасникам без повного декодування або мікшування. Завдяки цьому забезпечується наднизька затримка і зниження навантаження на сервер. Сучасні платформи розгортають SFU на граничних серверах, що сумісно з концепцією edge computing і дає змогу масштабувати групові сесії у 5G-мережі.

Таким чином, поєднання 5G, edge computing та ШІ-адаптації вимагає від мультимедійних протоколів більшої гнучкості: зменшувати латентність, ефективно використовувати широку смугу каналу та самостійно підлаштовуватися під мережеві умови. Окремі розширення (LL-HLS, CMAF, WebTransport, SRTP, SFU тощо) є відповіддю на ці вимоги, дозволяючи сучасним стрімінговим сервісам надавати високоякісний та масштабований

контент у режимі реального часу. Усі вищенаведені зміни інтегруються в загальну екосистему мультимедіа і узгоджуються з раніше описаними розділами роботи щодо архітектури та особливостей протоколів трансляції.

4.3. Рекомендації для інтеграції у сучасні платформи

У сучасних платформах стримінгові протоколи є невід'ємною складовою архітектури, що поєднує хмарні сервіси, контейнеризовані серверні рішення та глобальну мережу доставки контенту (CDN). Хмарні провайдери, такі як AWS, Microsoft Azure та Google Cloud, пропонують набори сервісів для обробки відеопотоків у режимі реального часу. Наприклад, AWS Elemental MediaLive та MediaPackage можуть приймати вихідні потоки (наприклад RTMP, SRT або WebRTC) з камер чи додатків, кодувати їх і пакетувати у формати HLS/DASH. Аналогічно Google Cloud Transcoder API дозволяє створювати адаптивні профілі трансляцій для різних пристроїв. Використання хмарних сервісів гарантує високу надійність і масштабованість, адже ресурси автоматично розподіляються залежно від навантаження.

Мережі доставки контенту (CDN), такі як Amazon CloudFront або Azure CDN, використовуються для кешування та реплікації відеоконтенту ближче до кінцевих користувачів. Наприклад, відеопотік може спочатку надходити на сервер у хмарі або локальний медіасервер (такий як Ant Media Server або Wowza) для транскодування і сегментування. Сформовані сегменти відео зберігаються у хмарному сховищі або безпосередньо передаються до вузлів CDN. Як наслідок, відеоконтент доставляється глядачам із мінімальними затримками і навантаження на основні сервери зменшується. У великих системах застосовують балансувальники навантаження і декілька регіональних медіасерверів для відмовостійкості та глобального покриття.

На стороні клієнта та сервера використовуються різні SDK і медіабібліотеки для забезпечення сумісності і взаємодії. Платформи Twilio та Agora надають WebRTC API і сервіси для організації відеодзвінків і стрімінгу

з мінімальною затримкою. Для відтворення потоків у мобільних додатках широко застосовують ExoPlayer (Android) та AVPlayer (iOS), які підтримують адаптивне відтворення HLS/DASH. У бекенд-системах використовують бібліотеку libSRT для надійної доставки потоків по SRT та FFmpeg для конвертування і мультипротокольного виходу відео. Наприклад, додаток може отримувати SRT-потік від зовнішньої камери, а сервер на основі FFmpeg перетворює його у HLS для доставки через CDN.

Забезпечення масштабованості та безперервності роботи системи передбачає автоматизовані рішення для моніторингу і відновлення потоків. Контейнерні оркестратори (наприклад, Kubernetes) можуть контролювати стан Wowza чи Ant Media Server та при необхідності перезапускати несправний сервіс. Крім того, горизонтальне масштабування дозволяє додавати нові екземпляри медіасерверів у години пікового навантаження (застосовують Kubernetes Horizontal Pod Autoscaler або аналогічні механізми у хмарних платформах). Для надійності також використовують одночасну реплікацію потоків на кількох CDN: такий підхід гарантує, що навіть у разі недоступності одного CDN кінцеві користувачі отримають контент через альтернативний канал.

На практиці такі інтегровані рішення вже застосовують у промислових масштабах. Наприклад, під час великої спортивної трансляції відеосигнал з камер надходить в AWS MediaLive, де кодується у кілька бітрейтів. Готові HLS-сегменти роздаються через Amazon CloudFront і інші CDN-провайдери, що забезпечує широку географічну доступність та стійкість до навантаження. У корпоративних рішеннях Wowza Server або Ant Media Server часто розгортають у Docker-контейнерах на кластерах Azure Kubernetes для одночасної обробки тисяч підключень. У мобільних додатках такі потоки відтворюють за допомогою ExoPlayer чи AVPlayer з адаптивним бітрейтом, що забезпечує плавне відтворення. Таким чином поєднання хмарних сервісів, контейнеризації, CDN і спеціалізованих SDK дозволяє побудувати надійну й масштабовану стримінгову платформу з автоматичним відновленням потоків.

4.4. Перспективи подальших досліджень

Постійне зростання попиту на потокове відео та розвиток технологій зумовлюють актуальність нових напрямків досліджень у цій галузі. Одним з них є автоматизований контроль якості користувацького досвіду (QoE). Наразі застосовуються алгоритми, які в реальному часі аналізують мережеві метрики — частоту буферизації, час завантаження та згасання кадрів — і виявляють артефакти зображення. Вбудовування машинного навчання (AI) дозволяє прогнозувати погіршення якості і заздалегідь коригувати параметри трансляції (наприклад, бітрейт або тип кодування). Таким чином адаптивні протоколи стають «розумнішими», оперативно реагуючи на зміни умов мережі для підтримки стабільної якості.

Новим трендом є гібридні транспортні рішення. Робочі групи IETF, зокрема MoQ (Media over QUIC), розробляють протокол, що поєднує в собі переваги QUIC і RTP, забезпечуючи одночасно низьку затримку та надійну двобічну передачу. Інтеграція WebRTC з CDN відкриває можливості масштабування: наприклад, передача відео через WebRTC може підтримуватися кешуванням сегментів у CDN або створенням перехідних вузлів (relay), що зменшує навантаження на джерело. Подібні рішення дозволяють використовувати WebRTC не лише для малих дзвінків, а й для широкомасштабних трансляцій, поєднуючи низьку латентність з можливостями CDN.

Перспективним є й використання P2P-технологій та мережевих проксі: глядачі можуть обмінюватися закешованими сегментами між собою через WebRTC Data Channel, зменшуючи навантаження на серверну інфраструктуру і скорочуючи затримки. Масштабування WebRTC до сотень і тисяч учасників вирішується шляхом впровадження SFU/MCU серверів, які динамічно розгортаються у хмарі для розподіленої обробки потоків. Однак такі UDP-орієнтовані протоколи ставлять нові завдання безпеки: необхідно забезпечити шифрування даних (зазвичай через DTLS/SRTP або SRT) і захист від

несанкціонованого доступу. У цьому контексті важливими є розробки стандартів безпеки та автентифікації для SRT, RIST і WebRTC.

Низьколатентні розширення HTTP-стрімінгу (LL-HLS, LL-DASH) також є об’єктом досліджень. Замість звичайних сегментів вони використовують дуже короткі фрагменти або чанки, що дозволяє досягти мінімальної затримки (порядку 2 секунд) і вимагає узгодження режимів обробки на CDN і плеєрах. Існують ініціативи з уніфікації таких технологій: наприклад, формат CMAF Low-Latency забезпечує сумісність між LL-HLS і LL-DASH. Водночас тривають роботи над стандартизацією нових протоколів і сервісів — Reliable Internet Stream Transport (RIST) як відкритого надійного транспорту, MoQ для відеоконференцій, а також протоколів WHIP/WHEP (WebRTC-HTTP Ingest/Egress Protocol), що спрощують підключення WebRTC-потоків через HTTP. Усвідомлення актуальності цих напрямів зумовлене тим, що сучасні сервіси мають одночасно забезпечувати високу якість, низьку затримку та захищеність передачі у великих розподілених мережах.

4.5 Висновки з розділу 4

У цьому розділі розглянуто оптимізацію вибору протоколів залежно від типу трансляції, вплив новітніх технологій та інтеграцію в сучасні стримінгові платформи.

Вибір протоколу залежить від вимог трансляції: HLS/DASH підходять для VoD і масштабних стрімів, WebRTC — для інтерактивного відео, SRT/QUIC — для умов із нестабільним з’єднанням та низькою затримкою. Технології 5G та edge computing знижують затримку, покращують стабільність і розширюють можливості масштабування. Хмарні сервіси та CDN забезпечують надійність і гнучкість стримінгових рішень, а бібліотеки на кшталт WebRTC і FFmpeg спрощують реалізацію. Перспективні напрями включають гібридні протоколи, AI-оптимізацію якості та безпечну передачу даних у великих системах.

Таким чином, ефективність мультимедійного стримінгу базується на грамотному виборі протоколу, адаптації до мережевих умов та інтеграції з сучасною інфраструктурою.

ВИСНОВКИ

У дипломній роботі здійснено комплексний аналіз протоколів передачі даних у мережах електронних комунікацій, орієнтованих на мультимедійне мовлення в реальному часі. Проведено класифікацію основних протоколів за типом транспорту, принципами доставки та рівнем адаптивності. Особливу увагу приділено сучасним UDP- та HTTP-орієнтованим рішенням, таким як SRT, QUIC, WebRTC, HLS та DASH, що є основою для стримінгових сервісів і платформ. Досліджено вплив новітніх технологій (5G, edge computing, AI-адаптації) на вимоги до транспортного рівня та поведінки протоколів у мережевому середовищі. У результаті розроблено рекомендації щодо інтеграції адаптивних протоколів у хмарну інфраструктуру із використанням CDN та SDK. Надано оцінку поточним викликам і перспективам подальших досліджень — зокрема, у напрямі протоколів нового покоління з вбудованими модулями штучного інтелекту, об’єднаних транспортних рішень (MoQ), гібридних потоків і масштабованого WebRTC.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Postel, J. (1981). RFC 793: Transmission Control Protocol. [Електронний ресурс] – Режим доступу: https://datatracker.ietf.org/doc/html/rfc793

2. Postel, J. (1980). RFC 768: User Datagram Protocol. [Електронний ресурс] – Режим доступу: https://datatracker.ietf.org/doc/html/rfc768

3. Ramakrishnan, K. K., Floyd, S., Black, D. (2001). The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168.

4. Iyengar, J., Thomson, M. (2021). RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport.

5. Pantos, R., May, N. (2017). HTTP Live Streaming (HLS). Apple Inc. [Електронний ресурс]

6. ISO/IEC 23009-1:2020. Information technology – Dynamic adaptive streaming over HTTP (DASH).

7. Ha, S., Rhee, I., et al. CUBIC: A New TCP-Friendly High-Speed TCP Variant – SIGOPS Operating Systems Review, 2008

8. Cardwell, N., Cheng, Y., Gunn, C., Yeganeh, S. H., Jacobson, V. (2016). BBR: Congestion-Based Congestion Control. Google Inc.

9. Google. WebRTC Native and JavaScript APIs. [Електронний ресурс] – https://webrtc.org/

10. Open Broadcaster Software (OBS) – [Електронний ресурс]: https://obsproject.com/

11. Medialooks. Low Latency Streaming Whitepaper. [Електронний ресурс]: https://www.medialooks.com

12. Ant Media Server – Офіційна документація. [Електронний ресурс]: https://antmedia.io

13. AWS Documentation. Amazon CloudFront Streaming. [Електронний ресурс]: https://docs.aws.amazon.com

14. Flussonic. SRT and QUIC Security Overview. [Електронний ресурс]: https://flussonic.com

15. StreamingMedia.com. Hybrid Streaming Architectures & WHIP/WHEP protocols. [Електронний ресурс]: https://www.streamingmedia.com

16. Arxiv.org – Публікації про машинне навчання для адаптації QoE. [Електронний ресурс]: https://arxiv.org

17. Gaol, F. L. (2012). Recent Progress in Data Engineering and Internet Technology. Springer.

18. GStreamer project – https://gstreamer.freedesktop.org

19. FFmpeg Documentation – https://ffmpeg.org/documentation.html