ВСТУП

Сучасні тенденції розвитку телекомунікаційних технологій та інтелектуальних систем спостереження зумовлюють зростання вимог до оперативності передавання відео та здатності систем реагувати на події в режимі реального часу. Відеонагляд перетворився з інструменту архівного аналізу на ключову складову концепцій «розумної безпеки», превентивного моніторингу, автоматизації та аналітики на основі штучного інтелекту. Водночас традиційні системи відеоспостереження часто характеризуються значною затримкою — від сотень мілісекунд до кількох секунд, що унеможливлює використання їх у критично важливих сценаріях.

Затримка відео має вирішальне значення для таких сфер, як керування безпілотними платформами, моніторинг промислових процесів, тактичні системи безпеки, медичні телероботизовані комплекси та системи реагування на інциденти. Навіть незначне збільшення латентності може призвести до порушення синхронізації, втрати контекстної інформації або зниження ефективності реагування. Саме тому у наукових дослідженнях та промислових розробках зростає попит на технології **low-latency відеонагляду**, здатні забезпечувати затримку менше 200 мс без втрати якості та стабільності.

Проблема низької затримки є комплексною, оскільки кожен етап формування відео — захоплення, кодування, транспортування, буферизація, декодування та відтворення — вносить свій внесок у загальну латентність. Для досягнення мінімально можливої затримки необхідно оптимізувати всі компоненти системи, а також правильно обрати транспортні протоколи, алгоритми кодування та мережеві механізми адаптації. Особливу роль відіграють сучасні технології реального часу, такі як WebRTC, SRT, QUIC, апаратне прискорення відеокодування та edge-обробка.

Актуальність теми також зумовлена інтенсивним розвитком ринку камер та серверних платформ, що пропонують вбудовані режими low-latency. Провідні виробники систем відеонагляду (Hikvision, Dahua, Axis, Hanwha, Uniview) впроваджують оптимізовані протоколи передавання та методи компресії, однак переважна більшість рішень не дає можливості налаштовувати систему під конкретні вимоги або інтегрувати їх у складні мережеві середовища. Це формує потребу у розробці власної архітектури, яка забезпечить контроль над усіма параметрами та дозволить побудувати масштабовану систему без критичних затримок.

Мета даної роботи полягає у розробці, дослідженні та практичній реалізації системи відеонагляду з низькою затримкою, здатної забезпечити стабільну доставку відеопотоку із загальною затримкою не більше 200 мс, гарантуючи при цьому необхідну якість зображення, стійкість до втрат пакетів та безпеку передавання.

Для досягнення мети були визначені такі основні завдання:

- провести аналітичний огляд сучасних технологій та протоколів відеоспостереження з акцентом на фактори, що впливають на затримку;
- виконати математичне моделювання ключових процесів відеопередачі та визначити межі мінімальної латентності;
- обґрунтувати вибір транспортної технології та кодувальних алгоритмів, які забезпечують low-latency режим;

- розробити архітектуру системи відеонагляду з низькою затримкою та визначити її функціональні та технічні вимоги;
- реалізувати прототип із використанням сучасних протоколів реального часу (WebRTC або SRT);
- провести тестування, аналіз результатів та оцінку відповідності проєктних параметрів заданим вимогам.

Об’єктом дослідження є процес передавання відеопотоків у мережах реального часу. Предметом дослідження — архітектурні та технологічні рішення, що забезпечують низьку затримку у системах відеонагляду.

Практична значущість роботи полягає у створенні архітектури та прототипу, які можуть бути використані як основа для впровадження низьколатентних систем спостереження в промислових, комерційних та критично важливих інфраструктурах.

Структурно робота складається з трьох розділів, у яких послідовно представлено аналітичний огляд, теоретичні моделі та проєктно-практичні результати.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ ВІДЕОНАГЛЯДУ

1.1. Архітектура сучасних систем відеоспостереження

Сучасні системи відеонагляду пройшли значну еволюцію — від аналогових CCTV-комплексів до високонавантажених IP-платформ, розподілених між периферійними, локальними та хмарними обчислювальними ресурсами. З огляду на це архітектура сучасної системи відеоспостереження вже давно перестала бути просто набором камер та моніторів. Тепер це багаторівнева інфраструктура, що включає інтелектуальні пристрої збору даних, мережеву інфраструктуру низької затримки, розподілені платформи обробки відео, аналітичні модулі, системи зберігання, а також сервіси моніторингу та керування доступом.

Архітектура відеонагляду практично завжди будується за модульним принципом, що дозволяє масштабувати систему відповідно до потреб, а також забезпечувати гнучкість у виборі технологій та апаратних рішень. У широкому розумінні сучасні системи ґрунтуються на таких ключових компонентах:

Периферійний рівень (edge layer) — камери та сенсори з локальною обробкою. Цей рівень відповідає за збір відеоданих та їх первинну обробку. Сучасні камери можуть мати вбудовані функції відеоаналітики, такі як розпізнавання об'єктів, детекція руху, підрахунок людей тощо. Це дозволяє зменшити навантаження на центральний сервер обробки та реагувати на події в реальному часі. Сенсори, такі як датчики руху, диму або температури, можуть бути інтегровані в систему для розширення її функціональності та забезпечення додаткової інформації про навколишнє середовище.

Транспортний мережевий рівень (network layer) — дротові або бездротові канали передачі відео. Цей рівень забезпечує передачу відеоданих від камер та сенсорів до центрального сервера обробки та зберігання. Вибір каналу передачі залежить від багатьох факторів, таких як відстань, пропускна здатність, вартість та вимоги до безпеки. Дротові канали, такі як Ethernet, забезпечують високу пропускну здатність та надійність, але можуть бути дорожчими та складнішими в установці. Бездротові канали, такі як Wi-Fi або 4G/5G, забезпечують більшу гнучкість та мобільність, але можуть бути менш надійними та мати обмежену пропускну здатність.

Рівень обробки та аналізу (processing layer) — сервери, відеореєстратори, платформи відеоаналітики. Цей рівень відповідає за обробку, аналіз та зберігання відеоданих. Сервери та відеореєстратори використовуються для запису та відтворення відео, а також для виконання базових функцій відеоаналітики. Платформи відеоаналітики надають більш розширені можливості, такі як розпізнавання облич, аналіз поведінки, виявлення аномалій тощо. Вони можуть бути реалізовані як програмне забезпечення, що встановлюється на сервер, або як хмарний сервіс.

Рівень зберігання (storage layer) — локальні та хмарні сховища. Цей рівень відповідає за зберігання відеоданих. Локальні сховища, такі як жорсткі диски або мережеві накопичувачі (NAS), забезпечують швидкий доступ до даних та більший контроль над ними. Хмарні сховища забезпечують більшу масштабованість, надійність та доступність, але можуть бути дорожчими та вимагати надійного інтернет-з'єднання. Вибір типу сховища залежить від багатьох факторів, таких як обсяг даних, вимоги до зберігання, бюджет та вимоги до безпеки.

Рівень керування та доступу (management layer) — програмні комплекси адміністрування. Цей рівень відповідає за керування системою відеонагляду, включаючи налаштування камер, управління користувачами,

перегляд відео, отримання сповіщень та формування звітів. Програмні комплекси адміністрування можуть бути реалізовані як веб-інтерфейси, десктопні програми або мобільні додатки. Вони повинні забезпечувати зручний та інтуїтивно зрозумілий інтерфейс для керування системою, а також забезпечувати безпечний доступ до даних.



Рис. 1.1 Модульна Архітектура Систем Відеонагляду

У цьому підрозділі розглянемо кожен з елементів архітектури та взаємозв’язки між ними більш детально, з особливим акцентом на вимоги до систем із низькою затримкою.

1.1.1. Периферійний рівень: джерела відеосигналу

Периферійний рівень є стартовою точкою формування відеопотоку. У сучасних системах використовуються такі типи камер:

IP-камери

Це найбільш поширене рішення. IP-камери формують цифровий потік безпосередньо всередині пристрою. Основні характеристики:

- підтримка кодеків H.264, H.265, H.265+, AV1, інколи — MJPEG;
- інтегровані процесори та акселератори апаратного кодування;
- можливість локальної аналітики (motion detection, face recognition, object tracking);
- передача по Ethernet, оптиці, Wi-Fi або 4G/5G.

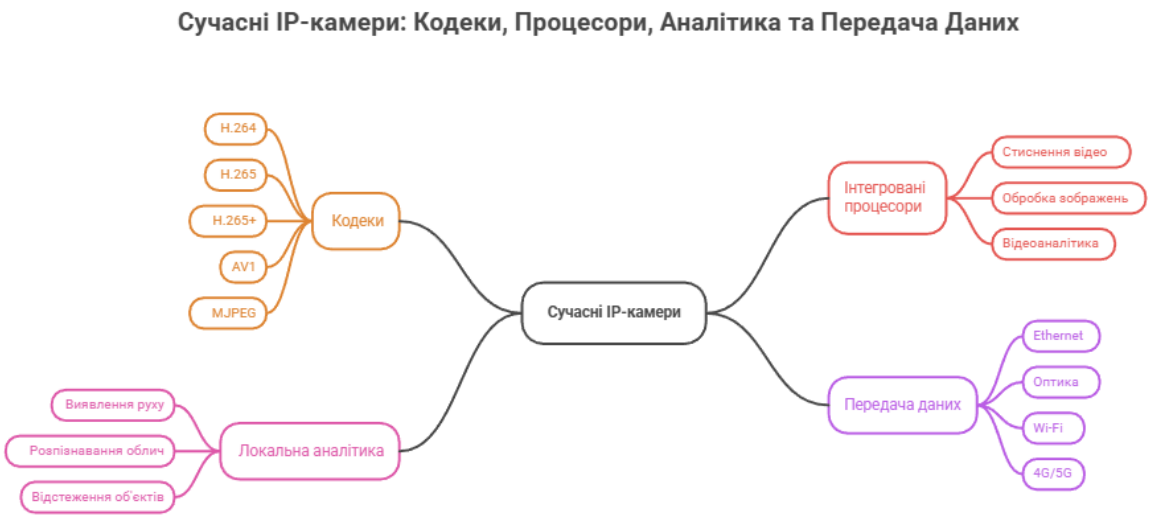


Рис. 1.2 Сучасні IP-камери: Кодеки, Процесори, Аналітика та Передача Даних

Панорамні та мультисенсорні камери

Забезпечують огляд 180° або 360° і використовуються для великих просторів. Частина камер має алгоритми stitching для зшивання зображень у реальному часі.

Камери з глибокою нейронною обробкою (AI-edge cameras)

Оснащені прискорювачами:

- NVIDIA Jetson,
- Google Coral,
- Intel Movidius,
- Huawei Ascend,
- ASIC- або FPGA-прискорювачами.

Це дозволяє виконувати аналітику саме на камері, зменшуючи затримку передачі та навантаження на сервери.

PTZ-камери (панорамування/нахил/зум)

Стосуються систем централізованого охоронного моніторингу. Керування PTZ потребує мінімальної затримки, адже навіть 300–500 мс лагу робить ручне наведення некерованим.

1.1.2. Мережева інфраструктура та комунікації

Надання низької затримки у відеосистемах значною мірою залежить від використаної мережевої технології.

Провідні канали (Ethernet):

- Fast Ethernet (100 Мбіт/с) — використовується у бюджетних системах.
- Gigabit Ethernet (1 Гбіт/с) — стандарт для сучасних проєктів.
- 2.5G / 5G / 10G Ethernet — застосовується у великих комплексах або для обробки потоків 4K/8K.
- PoE / PoE+ / PoE++ — забезпечують живлення камер через кабель.

Оптоволоконні мережі

Використовуються для передачі на великі відстані або для високих навантажень (особливо для 4K + AI-аналітики).

Переваги:

- мінімальні затримки (до кількох мікросекунд),
- імунітет до електромагнітних завад,
- дальність до десятків кілометрів.

Бездротові технології

Використовують там, де кабельне підключення неможливе.

- Wi-Fi 5/6/6E — середня затримка 5–20 мс.
- 5G — затримка до 1–5 мс у URLLC-сценаріях.
- Мікрохвильові лінки — для промислових систем у віддалених точках.

Для систем з низькою затримкою важливо забезпечити:

- мінімальну втрату пакетів,
- стабільну пропускну здатність,
- підтримку QoS та пріоритезацію відеотрафіку,
- зменшення джитеру.

1.1.3. Сервери, відеореєстратори та платформи обробки

Системи відеонагляду використовують різні підходи до обробки даних:

1. NVR (мережеві відеореєстратори)

Апартні пристрої з операційними системами реального часу, з оптимізацією під декодування/кодування відео. Підходять для малих та середніх систем.

Особливості:

- фіксована кількість каналів,
- локальне зберігання на HDD/SSD,
- обмежений набір AI-функцій.

2. Сервери відеообробки (VMS-server)

Працюють на базі:

- Linux / Windows Server,
- Docker-контейнерів,
- GPU-кластерів.

Підтримують:

- розподілене декодування,
- AI-відеоаналітику,
- балансування навантаження,
- низьколатентний стрімінг (WebRTC, RTSP 2.0, SRT).

3. Edge-сервери

Розміщуються ближче до камер та зменшують затримку за рахунок локальної обробки.

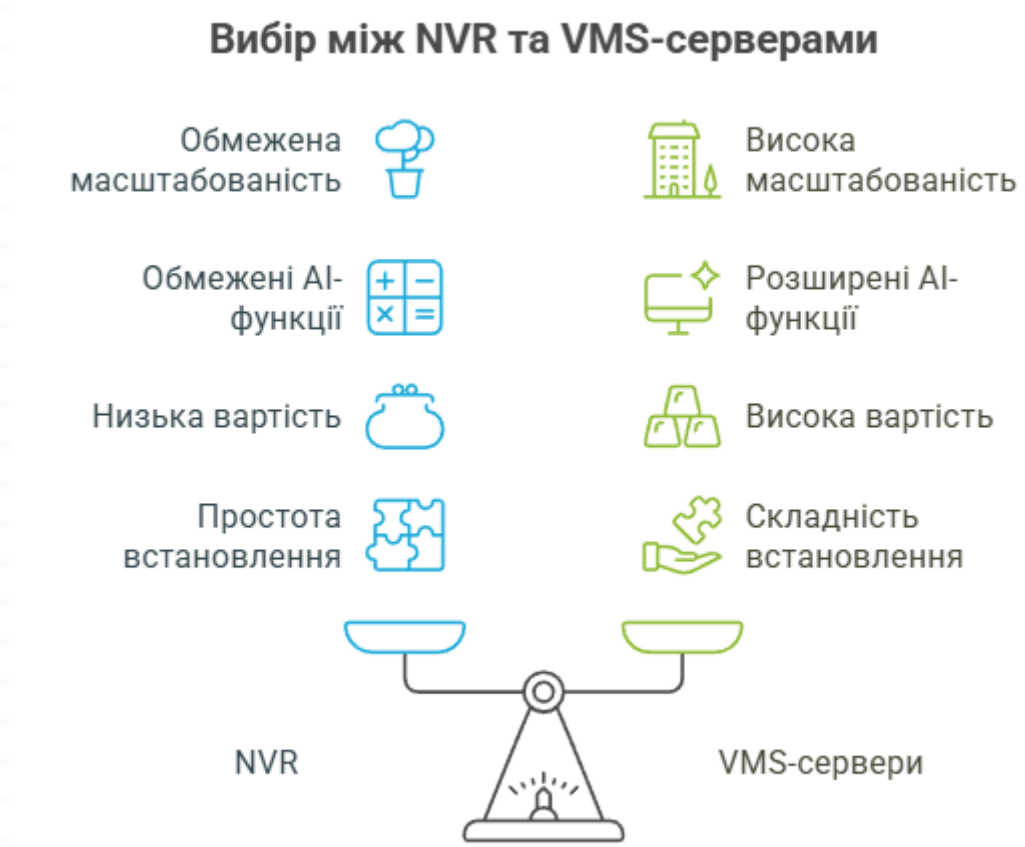


Рис 1.3 Вибір між NVR та VMS-серверами

Edge-сервери – це обчислювальні пристрої, які розміщуються ближче до джерела даних, тобто до камер. Вони виконують локальну обробку відеоданих, що дозволяє зменшити затримку та навантаження на мережу. Edge-сервери підходять для систем відеонагляду, де потрібна низька затримка, висока пропускна здатність та автономна робота.

1.1.4. Рівень зберігання відеоданих

Зберігання — критичний компонент системи.

Основні методи:

- **локальні HDD/SSD у камерах або NVR** — дешевше, але обмежено,
- **NAS/SAN сховища** — гнучкі, масштабовані,
- **хмарне зберігання (Cloud Video Storage)** — доступність та реплікація,
- **гібридне зберігання** — найсучасніша концепція.

Компресія відео

У системах з низькою затримкою пріоритет зміщується з максимального стискання до оптимального балансу між:

- якістю,
- ваговим розміром,
- швидкістю кодування.

Тому у багатьох випадках використовується:

- **H.265 Low-Latency Mode**,
- **H.264 Baseline**,
- **SRT Low-Latency**,
- **WebRTC VP8/VP9 для браузерів**.

1.1.5. Рівень керування, моніторингу та доступу

Центральним елементом архітектури є програмна платформа управління (VMS — Video Management System).

Основні функції:

- конфігурація камер;
- моніторинг у реальному часі;
- керування правами користувачів;
- ручне та автоматичне керування PTZ;
- робота з архівами;
- інтеграція з охоронними системами;
- розумні сценарії (automation scripts).

VMS може працювати за:

- централізованою моделлю — всі дані через головний сервер;
- розподіленою моделлю — обробка на edge-вузлах;
- хмарною моделлю — керування через browser-client.

У системах низької затримки використовуються протоколи:

- WebRTC — затримка 50–200 мс,
- SRT — адаптивна передача через нестабільні мережі,
- RTSP/RTP Low-Latency,
- HTTP/2 та HTTP/3 для сигналізації.

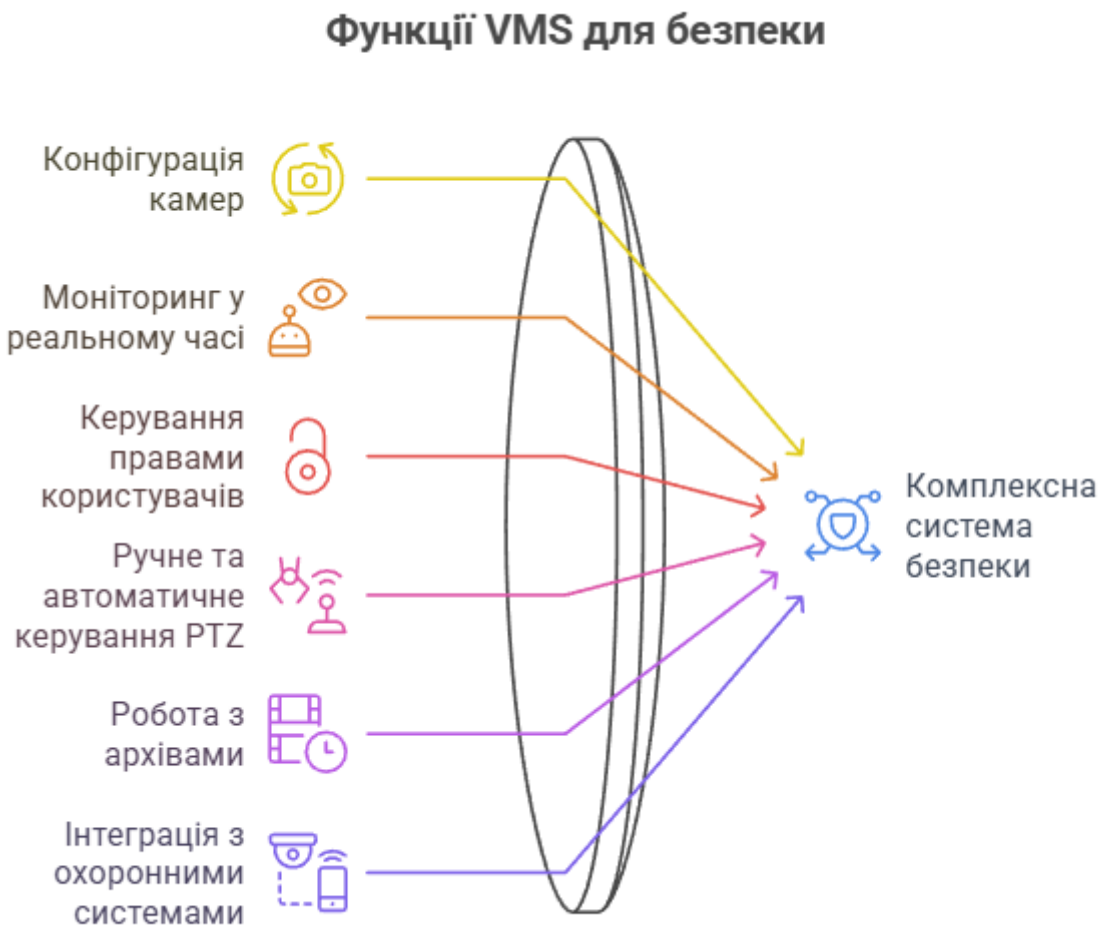


Рис.1.4 Функції VMS для безпеки

Вибір конкретного протоколу залежить від вимог до системи, характеристик мережі та можливостей обладнання.

1.1.6. Інтеграція з інтелектуальними модулями та автоматизація

Виявлення руху

Виявлення руху є однією з базових, але важливих функцій ШІ в системах відеонагляду. Традиційні системи виявлення руху часто реагують на будь-які зміни в кадрі, що призводить до великої кількості помилкових спрацювань, наприклад, через зміну освітлення, рух листя або тварин. ШІ дозволяє значно підвищити точність виявлення руху, фільтруючи ці помилкові спрацювання.

Як це працює:

- **Аналіз відеопотоку:** ШІ аналізує відеопотік у реальному часі, виявляючи зміни в пікселях.
- **Фільтрація шумів:** Алгоритми машинного навчання навчаються розрізняти реальні рухи від шумів, таких як зміни освітлення або погодні умови.
- **Налаштування чутливості:** Користувачі можуть налаштовувати чутливість виявлення руху, щоб адаптувати систему до конкретних умов.

Переваги:

- Зменшення кількості помилкових спрацювань.
- Підвищення ефективності моніторингу.
- Економія місця на носіях даних, оскільки записуються лише важливі події.

Визначення обличчя

Визначення обличчя – це технологія, яка дозволяє автоматично знаходити та ідентифікувати обличчя людей у відеопотоці. Ця функція широко використовується для контролю доступу, ідентифікації злочинців та інших цілей безпеки.

Як це працює:

- **Виявлення облич:** Алгоритми ШІ сканують відеопотік у пошуках облич.
- **Аналіз характеристик:** Після виявлення обличчя система аналізує його характеристики, такі як відстань між очима, форма носа та інші унікальні риси.
- **Порівняння з базою даних:** Отримані характеристики порівнюються з базою даних облич, щоб ідентифікувати особу.

Переваги:

- Автоматизація процесу ідентифікації.
- Підвищення рівня безпеки.
- Можливість швидкого пошуку осіб у відеоархіві.

Розпізнавання номерних знаків

Розпізнавання номерних знаків (ANPR) – це технологія, яка дозволяє автоматично зчитувати номерні знаки транспортних засобів. Ця функція використовується для контролю доступу на парковки, відстеження транспортних засобів та інших цілей.

Як це працює:

- **Виявлення номерного знака:** Алгоритми ШІ виявляють номерний знак на зображенні або відео.
- **Сегментація символів:** Система розділяє номерний знак на окремі символи.
- **Розпізнавання символів:** Алгоритми машинного навчання розпізнають кожен символ і перетворюють його на текст.

Переваги:

- Автоматизація контролю доступу.
- Підвищення ефективності відстеження транспортних засобів.
- Можливість інтеграції з іншими системами безпеки.

Класифікація об’єктів

Класифікація об’єктів – це функція, яка дозволяє системі відеонагляду розпізнавати та класифікувати різні об’єкти, такі як люди, транспортні засоби, тварини та інші. Це дозволяє більш точно реагувати на події та зменшує кількість помилкових спрацювань.

Як це працює:

- **Аналіз зображення:** Алгоритми ШІ аналізують зображення або відеокадр.
- **Виявлення об’єктів:** Система виявляє об’єкти на зображенні.
- **Класифікація об’єктів:** Алгоритми машинного навчання класифікують об’єкти за їх типом (людина, автомобіль, тварина тощо).

Переваги:

- Підвищення точності виявлення подій.
- Зменшення кількості помилкових спрацювань.
- Можливість налаштування реакції системи на різні типи об’єктів.

Прогнозування поведінки

Прогнозування поведінки – це більш складна функція ШІ, яка дозволяє системі відеонагляду аналізувати поведінку людей та прогнозувати можливі небезпечні ситуації. Наприклад, система може виявити підозрілу поведінку людини, яка довго стоїть біля банкомату, або групу людей, які збираються в одному місці.

Як це працює:

- **Аналіз поведінки:** Алгоритми ШІ аналізують рухи та дії людей.
- **Виявлення аномалій:** Система виявляє відхилення від нормальної поведінки.
- **Прогнозування ризиків:** На основі аналізу поведінки система прогнозує можливі ризики та попереджає про них.

Переваги:

- Запобігання злочинам та небезпечним ситуаціям.
- Підвищення рівня безпеки.
- Можливість швидкого реагування на потенційні загрози.

Виявлення покинутих предметів

Виявлення покинутих предметів – це функція, яка дозволяє системі відеонагляду автоматично виявляти предмети, залишені без нагляду. Це може бути корисним для запобігання терористичним актам, виявлення підозрілих предметів та інших цілей безпеки.

Як це працює:

- **Аналіз відеопотоку:** Алгоритми ШІ аналізують відеопотік у реальному часі.
- **Виявлення нерухомих об'єктів:** Система виявляє об'єкти, які тривалий час залишаються нерухомими.
- **Повідомлення про подію:** Якщо об'єкт залишається без нагляду протягом певного часу, система надсилає повідомлення про подію.

Переваги:

- Підвищення рівня безпеки.
- Запобігання терористичним актам.
- Можливість швидкого реагування на підозрілі предмети.

Аналітика черг – це функція, яка дозволяє системі відеонагляду аналізувати черги людей, визначати їх довжину, час очікування та інші параметри. Це може бути корисним для оптимізації роботи магазинів, банків та інших організацій, де є черги.

Як це працює:

- **Виявлення людей:** Алгоритми ШІ виявляють людей у відеопотоці.
- **Відстеження руху:** Система відстежує рух людей та визначає їх положення в черзі.
- **Аналіз параметрів черги:** Система аналізує довжину черги, час очікування та інші параметри.

Переваги:

- Оптимізація роботи організації.
- Підвищення задоволеності клієнтів.
- Можливість швидкого реагування на зміни в черзі.

Інтеграція штучного інтелекту в системи відеонагляду значно розширила їх функціональні можливості та підвищила ефективність. Функції, такі як виявлення руху, визначення обличчя, розпізнавання номерних знаків, класифікація об'єктів, прогнозування поведінки, виявлення покинутих предметів та аналітика черг, дозволяють автоматизувати процеси моніторингу, підвищити рівень безпеки

1.1.7. Особливості архітектури систем із низькою затримкою

Стандартна архітектура відеонагляду не орієнтована на мінімізацію лагу. Для систем низької затримки потрібні додаткові вимоги:

1) Обробка наближена до джерела (edge-processing)

Частина аналітики, передкодування, фільтрації шумів та стабілізації повинна виконуватися на камері або на edge-сервері, мінімізуючи передачу сирих даних.

2) Протоколи низької затримки

Обов’язкове використання:

- WebRTC,
- Low-Latency HLS,
- SRT,
- UDP-RTP.

3) Мережі з QoS та пріоритетом відеопакетів

Застосування VLAN, DSCP-маркування, Traffic-Shaping.

4) Використання апаратного прискорення

GPU, FPGA, ASIC для:

- енкодингу,
- декодування,
- обробки потоків.

5) Скорочення буферизації

Зменшення:

- receive buffer,

- jitter buffer,
- GOP size.

1.2. Джерела затримок у відеопотоках

Затримка (latency) у системах відеоспостереження — це час, що проходить між моментом захоплення кадру камерою та відображенням цього кадру на кінцевому пристрої. Для стандартних систем спостереження затримка у межах 1–3 секунд є прийнятною. Проте у комплексах оперативного реагування, безпілотних системах, керуванні PTZ-камерами або у безпекових сценаріях із аналітикою реального часу навіть 300–500 мілісекунд можуть стати критичними.

Затримка у відеопотоці формується сумою кількох компонентів, кожен з яких залежить від обладнання, алгоритмів та технологій передачі даних. У цьому підрозділі розглянуто основні етапи формування відеосигналу та їхній вплив на загальний час доставки кадру.

1.2.1. Затримки під час захоплення кадру

Захоплення (capture) — перший етап обробки відео. Він включає:

- зчитування світлового сигналу CMOS/CCD сенсором,
- оцифрування аналогового сигналу,
- формування кадру у пам’яті пристрою.

Чинники, що створюють затримку:
1) Експозиція (shutter time)

Сенсор потребує часу на накопичення світла. Для стандартних камер:

- 1/30 – 1/60 секунди для стандартного освітлення,

- 1/120 – 1/1000 секунди для швидких сцен.

Проблема: чим довша експозиція, тим більша затримка.

2) Зчитування сенсора (sensor readout)

CMOS-сенсори використовують **rolling shutter**, що означає покадрове зчитування рядків.
Типова затримка: **5–20 мс**.

3) Попередня обробка в камері

- корекція кольору,
- шумозаглушення,
- стабілізація зображення,
- HDR/WDN обробка.

Це додає ще **2–10 мс**.

1.2.2. Кодування та транскодування відео

Кодування — один з найбільш «важких» етапів формування затримки.

Таблиця 1

Особливості впливу кодеків:

Кодек	Середня затримка	Особливості
H.264	20–50 мс	Швидкий, добре підтримується
H.265	40–120 мс	Ефективніший, але важчий для обробки
MJPEG	5–10 мс	Мінімальна затримка, але великий бітрейт
VP8/VP9	20–80 мс	Часто використовується у WebRTC
AV1	100–200+ мс	Дуже ефективний, але повільний

Причини затримки при кодуванні:

1. Створення груп кадрів (GOP):

   Якщо GOP=30, затримка збільшується, бо ключові кадри (I-frames) створюються рідко.
2. Буферизація перед кодуванням.

   Камера накопичує кілька кадрів, щоб оптимізувати стиснення.
3. Транскодування на стороні сервера.

   Перекодування у інший формат (наприклад, RTSP → WebRTC) може додавати 50–200 мс.
4. Використання софтверного кодування (CPU), а не апаратного (ASIC/GPU).



Рис 1.3 Відеокодування

Приклад:

Кодування відео 1080p за допомогою CPU може зайняти значно більше часу, ніж кодування за допомогою GPU. Це призводить до збільшення затримки та навантаження на систему.

Рекомендації:

Для мінімізації затримки слід використовувати апаратне кодування, якщо це можливо. ASIC та GPU спеціально розроблені для обробки відео та можуть значно прискорити процес кодування. Це дозволяє зменшити затримку та покращити продуктивність системи.

Затримка відео є важливим фактором, який слід враховувати при розробці систем передачі відео в реальному часі. Розмір GOP, буферизація перед кодуванням, транскодування на стороні сервера та використання софтверного кодування – це основні фактори, які впливають на затримку. Оптимізація цих факторів дозволяє мінімізувати затримку та покращити користувацький досвід.

1.2.3. Транспортні протоколи (RTSP, RTMP, WebRTC, SRT, HLS, DASH)

Тип протоколу критично впливає на кінцеву затримку. Різні протоколи мають різну архітектуру передачі та різну чутливість до втрат пакетів.

RTSP/RTP

- В середньому: **150–500 мс**.
- Працює поверх UDP.
- Мінімальна буферизація.
- Одне з найкращих рішень для low-latency у закритих мережах.

RTMP

- Затримка: **1–3 секунди**.
- Створений для Flash.
- Використовується у старих VMS або стримінгових платформах.
- Потребує TCP → збільшує затримку.

WebRTC

- Затримка: **50–200 мс**.

- Оптимізований для реального часу.
- Використовує STUN/TURN.
- Мінімізує буфери, працює через UDP.
- Ідеальний для систем із низькою затримкою.

SRT (Secure Reliable Transport)

- Затримка: **80–500 мс**, залежно від:
  - налаштувань jitter buffer,
  - втрат пакетів,
  - пропускної здатності.

SRT адаптивний, добре працює через нестабільні мережі (LTE, 5GN, інтернет).

HLS (HTTP Live Streaming)

- Типова затримка: **6–30 секунд**.
- Працює на сегментуванні відео (TS-сегменти по 2–6 сек).
- Використовується для публічного стримінгу, але не підходить для low-latency.

Low-Latency HLS (LL-HLS)

- Затримка: **1–2 секунди**.

MPEG-DASH

- Затримка: **2–10 секунд**.
- Також сегментний підхід.

1.2.4. Буферизація відеопотоку

Буферизація (buffering) — один із найбільших джерел затримки.

Види буферів:

Jitter buffer (мережевий буфер)

Потрібен для вирівнювання нерівномірного надходження пакетів. Стандартні значення:

- 30–150 мс (real-time),
- 300 мс у стабілізаційних сценаріях.

Decoder buffer

Кожний декодер (особливо H.265) має власний буфер.

Player buffer

Браузери та мобільні плеєри часто роблять додаткове кешування.

Rebuffering

При втраті пакетів відбувається накопичення кадрів → затримка може різко збільшуватися до секунд.

1.2.5. Затримки при серверній обробці

Сервер може виконувати багато операцій, що впливають на latency:

Основні джерела затримок:

Аналіз відео (AI/ML):

- виявлення об’єктів,
- класифікація,

- розпізнавання номерів.

Типова затримка:

- GPU: 5–15 мс,
- CPU: 50–200 мс.

Транскодування у різні формати:

Наприклад:

- H.265 → WebRTC,
- RTSP → HLS,
- MJPEG → VP8.

Кешування для балансу навантаження.

Передача у кілька потоків (multicasting, simulcasting).

1.2.6. Декодування на клієнтській стороні

Декодування відео відбувається на:

- браузерах,
- мобільних пристроях,
- VMS-клієнтах,
- вбудованих пристроях.

Джерела затримок:

1) Розмір буфера декодера

Браузери (Chrome/Firefox/Safari) часто створюють приховану буферизацію, яка може додавати **100–300 мс**.

2) Тип декодера:

- апаратний → 5–15 мс,
- програмний → 20–50 мс,
- H.265 складніший для декодування.

3) Формування кадру на дисплеї

Затримка оновлення екрану (display refresh):

- 60 Гц → 16.6 мс,
- 120 Гц → 8.3 мс.

1.2.7. Сукупна затримка відеопотоку

Таблиця 1.2

Наведемо типовий приклад для стандартної IP-камери (H.264 + RTSP):

Етап	**Затримка**
Захоплення	10–30 мс
Кодування	20–50 мс
Мережа (LAN)	1–5 мс
Протокол (RTSP)	50–150 мс
Jitter buffer	30–120 мс
Серверна обробка	10–50 мс
Декодування та відтворення	20–40 мс
Загальна затримка	**150–400 мс**

Для WebRTC затримки зменшуються до:

50–200 мс, залежно від конфігурації.

1.3. Огляд протоколів передавання відео з точки зору затримки

Вибір транспортного протоколу суттєво впливає на затримку у системах відеоспостереження. Навіть за ідентичних умов — однакова камера, однакова мережа — різні протоколи дають різні результати. Це пов’язано з особливостями доставки пакетів, типом сегментації, буферизацією, корекцією втрат, принципами роботи на транспортному рівні та використаними механізмами синхронізації.

У цьому підрозділі проведено порівняльний аналіз популярних протоколів передавання відео — **RTSP, RTMP, WebRTC, SRT, HLS**, — кожен з яких займає свою нішу та кожен має різний рівень затримки, масштабуємості та надійності.

1.3.1. RTSP (Real Time Streaming Protocol)

RTSP — один із найпоширеніших протоколів у професійних системах відеоспостереження. Він використовується камерами, відеореєстраторами та VMS у більшості корпоративних мереж.

Архітектурні особливості

- передавання відео здійснюється через **RTP поверх UDP або TCP**;
- мінімальна буферизація;
- низькі накладні витрати протоколу;

- можливість прямої доставки від камери до клієнта без серверного ретранслятора;
- підтримка у більшості камер (ONVIF, Hikvision, Dahua, Axis).



Рис 1.4 Вимоги до протоколу передачі відео

Типові джерела затримки:

- внутрішні буфери камери (25–80 мс),
- мережевий jitter buffer (50–150 мс),
- декодування (15–30 мс).

Типова затримка RTSP:

150–400 мс у локальній мережі (LAN)

300–1000 мс через Інтернет

Переваги:

- низька затримка,
- стабільність у LAN,
- підтримка мультикасту,
- мінімальне навантаження на сервер.

Недоліки:

- проблеми з NAT/Firewall,
- погана робота у веббраузерах (потрібен проксі/перекодування),
- відсутність адаптивного бітрейту.



Рис 1.5 Зважування переваг і недоліків UDP для потокового відео

UDP є корисним протоколом для потокового відео в певних сценаріях, особливо там, де важлива низька затримка та стабільність у локальній мережі. Однак, його недоліки, такі як проблеми з NAT/Firewall, погана підтримка у веббраузерах та відсутність адаптивного бітрейту, обмежують його застосування в більш широкому контексті. При виборі протоколу для потокового відео необхідно враховувати всі ці фактори та обирати протокол, який найкраще відповідає конкретним вимогам та умовам трансляції. У багатьох випадках, використання протоколів на основі TCP, таких як HLS або DASH, може бути більш прийнятним рішенням, особливо для трансляцій через Інтернет.

1.3.2. RTMP (Real-Time Messaging Protocol)

RTMP був основним стандартом для живих трансляцій у часи Adobe Flash, і досі використовується стрімінговими платформами та старішими VMS.

Архітектурні особливості

- працює **поверх TCP**, що вже створює додаткову затримку через механізми підтвердження пакетів;
- має значні внутрішні буфери для гарантії доставки;
- використовує власний формат контейнера FLV.

Типові затримки:

Затримка RTMP становить: **1–3 секунди у стабільних мережах до 5 секунд у мобільних мережах або при втраті пакетів**

Переваги:

- висока сумісність зі стрімінговими платформами,
- стабільність у мережах TCP,
- простота налаштування.

Недоліки:

- велика затримка,
- протокол застарів,
- вимагає серверного ретранслятора (Nginx RTMP, Wowza),
- не підтримується сучасними браузерами.

RTMP практично не застосовується в системах оперативного відеонагляду з низькою затримкою.

1.3.3. WebRTC (Web Real-Time Communication)

WebRTC — найсучасніший протокол для передачі відео з низькою затримкою. Спочатку створений для відеодзвінків, сьогодні використовується у професійному обладнанні, хмарних відеосервісах та системах відеонагляду.

Архітектурні особливості

- працює через **UDP / SRTP**;
- підтримує NAT traversal (STUN/TURN);
- автоматично адаптує бітрейт, FPS, роздільність;
- використовує кодеки VP8, VP9, H.264, Opus.

Типові джерела затримки:

- агресивна мінімізація буферів,
- оптимізований стек для low-latency,
- дуже швидке доставлення пакетів.

Типова затримка WebRTC:

50–200 мс у LAN

100–300 мс у глобальних мережах

Переваги:

- найменша затримка серед усіх масових протоколів,
- підтримка браузерами без плагінів,
- безпечний (DTLS, SRTP),
- автоматичний контроль якості (відновлення після втрат, A/V sync).

Недоліки:

- складність налаштування (потребує SFU-сервера),
- велике навантаження на процесор під час транскодування,
- не підтримує високу затримку мереж (на відміну від SRT).

WebRTC є найбільш відповідним протоколом для систем відеонагляду з низькою затримкою.

1.3.4. SRT (Secure Reliable Transport)

SRT — сучасний протокол, створений компанією Haivision, оптимізований для нестабільних мереж, таких як мобільний інтернет, супутниковий зв’язок чи міжнародні магістралі.

Архітектурні особливості

- базується на UDP,
- має механізми:
    - автоматичної корекції втрат,
    - ARQ (повторна передача пакетів),
    - адаптації до змін мережі,
    - шифрування (AES-128/256),
- забезпечує надійну доставку навіть при втраті 20–25% пакетів.

Типова затримка SRT:

80–500 мс, залежно від:

- налаштувань буфера (latency mode),
- Packet Loss Recovery,
- пропускної здатності каналу.

Переваги:

- висока якість відео через нестабільний канал,
- придатний для віддалених трансляцій,
- краща стійкість ніж RTSP/RTMP.

Недоліки:

- більша затримка, ніж у WebRTC,
- потрібна конфігурація буферів,
- не підтримується браузерами без адаптерів.

1.3.5. HLS (HTTP Live Streaming)

HLS — стандарт Apple для масштабованого стримінгу. Він не призначений для низької затримки.

Архітектурні особливості

- відео ділиться на сегменти (TS/MP4) тривалістю 2–10 секунд,
- клієнт завантажує сегменти HTTP-запитами,
- кешування CDN збільшує затримку,
- працює виключно через TCP.

Типові затримки HLS:

6–30 секунд (класичний варіант) **2–5 секунд для Low Latency HLS (LL-HLS)**

Переваги:

- найкраще масштабування у світі,
- простота роботи через CDN,
- підтримка усіма браузерами.

Недоліки:

- непридатний для систем відеоспостереження,
- великий час початку відтворення,
- сегментація створює затримку незалежно від мережі.

1.3.6. Порівняльний аналіз протоколів за затримкою

Таблиця 1.2

типові затримки

Протокол	**Типова затримка**	**Мінімальна можлива**	**Стійкість до втрат**	**Призначення**
WebRTC	**50–200 мс**	~30 мс	середня	відеодзвінки, low-latency відеонагляд
RTSP (UDP)	150–400 мс	~80 мс	низька	локальні мережі, VMS
SRT	80–500 мс	~70 мс	висока	нестабільні мережі, LTE/5G
RTMP	1–3 с	~0.8 с	середня	стримінг, старі системи

HLS	6–30 с	~2 с (LL-HLS)	висока	масштабні онлайн-трансляції

1.3.7. Висновки щодо вибору протоколу

- **Найнижча затримка: → WebRTC**
- **Найкраща якість у нестабільних мережах: → SRT**
- **Найкращий вибір у LAN-системах: → RTSP**
- **Для публічних трансляцій: → HLS**
- **Для сумісності зі старими VMS: → RTMP**

Для систем відеоспостереження з вимогами низької затримки оптимальним протоколом є **WebRTC**, тоді як **RTSP** забезпечує хороші результати у локальних мережах. **SRT** підходить для ситуацій, де сигнал передається на великі відстані або через нестабільні канали.



RTMP слід використовувати лише для сумісності зі старими системами управління відео (VMS). Для нових проектів рекомендується використовувати інші протоколи, такі як HLS, SRT або WebRTC.

1.4. Компроміси «затримка — якість — стабільність»

Розробка низьколатентних систем відеонагляду передбачає одночасне досягнення трьох критичних параметрів: **мінімальної затримки**, **високої якості відеопотоку** та **стабільності передачі** в умовах різної пропускної здатності мережі. Однак у реальних системах ці цілі конфліктують між собою, що змушує застосовувати збалансовані технічні підходи та визначати пріоритети залежно від сценарію використання. Компроміс між зазначеними характеристиками є фундаментальною проблемою потокового передавання даних, оскільки поліпшення одного параметра часто призводить до погіршення інших.

1.4.1. Взаємозв’язок між якістю відео та затримкою

Якість відеопотоку значною мірою залежить від:

- роздільної здатності;
- частоти кадрів;
- ступеня стиснення;
- складності кодека та вибору профілю;
- бітрейту.

Для досягнення високої якості кодувальник виконує складні операції аналізу руху, прогнозування та компенсації шумів, що суттєво збільшує час обробки. Використання високоефективних кодеків (H.265/HEVC, AV1) забезпечує значно кращу якість при нижчому бітрейті, але потребує тривалішого часу на енкодинг, що прямо збільшує затримку.

Відеонаглядові системи, орієнтовані на мінімальну латентність, переважно застосовують:

- **спрощені профілі кодеків** (Baseline, Main для H.264);
- **мінімальну кількість B-кадрів або їх повну відмову**;
- **підвищену частоту ключових кадрів (I-frames)**;
- **апаратне кодування** замість програмного.

Це дозволяє суттєво зменшити час обробки кадру, хоча у деяких випадках відео стає менш деталізованим та більш чутливим до шуму.

1.4.2. Вплив стабільності мережі на затримку

Стабільність відеопотоку визначається:

- рівнем втрати пакетів;
- варіацією затримки (jitter);
- пропускною здатністю каналу;
- кількістю стрибків RTT;
- навантаженням на мережевий вузол.

Для забезпечення стабільності системи зазвичай використовують адаптивну буферизацію. Але збільшення розміру буфера лінійно підвищує затримку. У низьколатентних системах буферизація обмежується мінімальними значеннями або повністю динамічно регулюється.

Класичний компроміс виглядає так:

- **Великий буфер** → висока стабільність, низькі втрати, але велика затримка.
- **Малий буфер** → наднизька затримка, але можливі ривки (stutters), артефакти та розриви потоку.

Технології Error Correction, такі як **ARQ**, **FEC**, **Hybrid FEC**, здатні підвищити стабільність, але також збільшують затримку через додаткові службові пакети та повторні запити.

1.4.3. Компроміс між якістю та стабільністю передачі

Забезпечення стабільності можливе завдяки зменшенню бітрейту та адаптації відеопотоку до доступної пропускної здатності. Але зниження бітрейту завжди веде до деградації якості:

- з’являються артефакти блокування;
- знижується чіткість;
- погіршується деталізація при швидких сценах.

Адаптивні алгоритми потокового передавання, такі як **ABR (Adaptive Bitrate)** у DASH/HLS або внутрішня адаптація в SRT/WebRTC, активно регулюють бітрейт. Проте швидкість адаптації визначає компроміс:

- **швидка адаптація** → менше розривів, але можливі різкі зміни якості;
- **повільна адаптація** → стабільніша якість, але більший ризик буферизації.

1.4.4. Затримка як критичний обмежувальний фактор

Для низьколатентних систем відеоспостереження першочерговим параметром є саме **мінімальна затримка**, тому інженери змушені жертвувати частиною якості та стабільності, обираючи:

- простіші алгоритми кодування;
- меншу глибину черг;
- найменші допустимі буферні межі;
- протоколи прямої передачі (WebRTC, SRT), а не сегментовані (HLS/DASH);
- обмеження на бітрейт для уникнення перевантаження каналу.

У такій конфігурації система може працювати з затримкою менше 200 мс, але якість може бути нижчою, особливо у складних сценах.

1.4.5. Балансування трьох параметрів у різних сценаріях

Відеоспостереження за периметром або приміщеннями

- Пріоритет: якість → стабільність → затримка.
- Дозволена затримка: 0.5–2 с.
  Часто використовується RTSP або RTMP.

Відеонагляд у транспортних системах (дрони, автономні роботи)

- Пріоритет: затримка → стабільність → якість.
- Максимально допустима затримка: 50–200 мс.
  Типові протоколи: WebRTC, SRT.

Охорона об’єктів у режимі реального часу

- Пріоритет: стабільність → затримка → якість.
- Затримка: 200–500 мс, але без розривів.
  Алгоритми адаптації та пом’якшення втрат.

Передавання архівного відео або оглядових потоків

- Пріоритет: якість → стабільність → затримка. Затримка не відіграє критичної ролі.

Таким чином, жодна сучасна система не може одночасно забезпечити найвищу якість, нульову затримку та абсолютну стабільність — оптимізується лише два із трьох параметрів.

1.4.6. Технологічні підходи до мінімізації компромісів

Сучасні системи з низькою затримкою застосовують комплекс технічних рішень, які дозволяють частково нівелювати компроміси:

- **апаратне кодування** (NVENC, QuickSync, ASIC-кодери);
- **low-latency режими кодеків**;
- **протоколи зі зворотним зв'язком у реальному часі (WebRTC, SRT)**;
- **динамічне керування бітрейтом**;
- **використання UDP-транспорту замість TCP** для швидкого відновлення при втратах;
- **дискретизація кадрів** для пропуску надлишкових сцен.

Такі методи дозволяють досягти балансу, де затримка мінімізується до прийнятного рівня, а якість і стабільність залишаються достатніми для конкретного завдання.

Компроміс «затримка — якість — стабільність» є фундаментальним інженерним обмеженням, яке визначає архітектуру будь-якої системи відеонагляду. Для досягнення низької затримки необхідно спростити кодування, зменшити буферизацію та застосовувати спеціалізовані

транспортні протоколи. Проте кожне з цих рішень впливає на якість та стабільність передачі, тому вибір оптимальної конфігурації залежить від сценарію використання системи.

1.5. Існуючі платформи та рішення з низькою затримкою

1.5.1. Традиційні комерційні бренди та їхні системи — Hikvision, Dahua, Axis

Hikvision

- Hikvision — один з глобальних лідерів у сфері систем відеонагляду; їх IP-камери, NVR та VMS широко використовуються як у побутовому, так і корпоративному секторах.
- Камери підтримують сучасні кодеки (H.264/H.265), мережеве під’єднання, PoE, можливість підключення до NVR або серверів, що робить їх гнучкими.
- Також деякі моделі мають вдосконалені сенсори для нічного бачення (наприклад, технологія, відома як Darkfighter — вона дає змогу у складному освітленні, вночі, отримувати кольорове відео, що підвищує якість зображення.
- Проте, як показує аналіз ринку й огляди, класичні системи від Hikvision (як і інших брендів) не завжди оптимізовані під low-latency: вони орієнтовані на стабільність, зберігання відео, широку функціональність, а не на мінімальну затримку.
- У ситуаціях, коли потрібно швидке реагування, оперативне відео з мінімальним лагом, стандартні IP-камери + NVR від Hikvision можуть демонструвати помітну затримку — особливо якщо використовується

запис, архів, перегляд через інтернет, або якщо додається відеоаналітика та обробка на сервері.

Dahua Technology

- Dahua — ще один великий гравець на ринку IP-відеоспостереження. Їхні камери і рішення широко використовуються через співвідношення ціна / функціональність / масштабованість.
- Dahua, подібно до Hikvision, орієнтована на комплексні рішення: камери, NVR, VMS, інтеграцію, зберігання відео, архіви, аналітику.
- Для багатьох задач (архів, аналітика, запис, охорона) така інфраструктура є оптимальною. Але, як і в випадку з Hikvision, класичні системи Dahua не гарантують мінімальну затримку — є компроміс між функціональністю, якістю та latency.

Axis Communications

- Axis — бренд, який фігурує серед лідерів ринку IP-відеокамер та систем.
- Часто Axis обирають за якість, надійність, сумісність зі стандартами, але в оглядах можливостей низької латентності для Axis часто згадується, що при порівнянні з деякими конкурентами відео може бути менш «швидким» або з більшим часом обробки, особливо при високій роздільній здатності.
- Таким чином, як і інші великі бренди, Axis підходить для стабільних, масштабованих систем, де priority — якість і надійність, а не найнижча можлива затримка.

1.5.2. WebRTC та сучасні low-latency рішення

Останніми роками дедалі популярнішими стають рішення для відеоспостереження, які базуються на протоколах і підходах, розроблених

для реального часу — такі як WebRTC, та інші подібні системи, які орієнтовані на мінімальну затримку й швидку доставку відео.

Сучасні підходи з edge-обробкою та low-latency стрімінгом

- Є наукові та інженерні розробки, які поєднують edge-обладнання (камера або пристрій на місці) + адаптивну буферизацію + легке кодування/декодування, щоб досягти наднизької латентності та пристойної точності аналізу. Наприклад, в одному з досліджень показано, що при оптимізованій архітектурі можна досягнути **end-to-end latency < 50 ms** при розпізнаванні на edge-пристрої. arXiv
- Такі системи підходять для задач із жорсткими вимогами до часу реагування — дрони, контроль руху, охорона, автоматичне реагування, real-time аналітика.

Міжплатформні middleware / VMS, що підтримують multi-протоколи й low-latency

- Деякі програмні платформи (middleware) підтримують одночасне підключення камер/пристроїв від різних брендів (наприклад, Hikvision, Dahua, Axis та ін.), і дають змогу «перепаковувати» потоки, ретранслювати їх, адаптувати під WebRTC/RTSP/SRT з метою зменшення затримки та забезпечення сумісності. daxunview.com
- Це дає змогу модернізувати існуючі інсталяції (з класичними IP-камерами), не міняючи все обладнання, але додавши шар, який підтримує low-latency стрімінг для клієнтів або локального моніторингу.

1.5.3. Переваги та обмеження різних підходів — з огляду на задачу низької затримки

Підхід / Платформа	**Переваги**	**Обмеження / недоліки (для low-latency)**
Hikvision / Dahua / Axis (традиційні IP + NVR/VMS)	Надійність, масштабованість, аналітика, архів, великий вибір обладнання	Затримка через запис/архів, буферизацію, сервера, менш підходять для real-time задач
WebRTC-базовані рішення / Edge-системи	Мінімальна латентність (<100–200 мс), швидка доставка, real-time	Можливі компроміси на бітрейті, якості, стабільності; складніша архітектура
Middleware / гібридні системи (старе + нове)	Гнучкість, сумісність, можливість модернізації, зменшення latency без заміни камер	Додатковий шар обробки, потребують налаштування, можуть з’явитися точки відмови

1.5.4. Висновки та рекомендації для low-latency систем відеонагляду

- Якщо мета — **архів, безпека, потрібна висока якість зображення, аналітика, зберігання** — класичні системи на базі Hikvision, Dahua, Axis є оптимальними.
- Якщо потрібне **реальне відео з мінімальною затримкою**, оперативність, негайна реакція — найкращим вибором будуть **WebRTC-системи або modern edge-рішення**, або комбіновані системи з middleware, які перетворюють класичні IP-потоки у low-latency стріми.
- Для проектів, де потрібен «золото-середній» баланс між якістю, надійністю й затримкою — має сенс використовувати гібридні системи: камери від перевірених брендів + спеціалізоване ПЗ / медіасервер / перетворення потоків.

Короткі рекомендації (вибір залежно від сценарію)

- Якщо потрібен **безпечний професійний енкодер** з гранично стабільним низьким енкодинг-латенті — розглядайте **Haivision Makito X** (апаратний енкодер).
- Для **власного медіасервера / інтерактивних застосунків** (дрони, керування, телемедицина) — **Ant Media**, **OvenMediaEngine** або **Wowza** з WebRTC/LL-HLS/WHIP підтримкою дають добрий баланс latency ↔ масштаб.
- Для **масштабованих, глобальних, інтерактивних платформ** розгляньте **Red5 Pro** або CDN-ready WebRTC провайдери (Dolby.io / Millicast).
- Якщо у вас **існуючі IP-камери** (Hikvision/Dahua/Axis) — найшвидший шлях до низької затримки — додати **WebRTC-gateway / media server** (go2rtc, Janus, AntMedia, OvenMediaEngine) і/або налаштувати на камерах low-latency режими (Axis має офіційний low-latency mode).

1.6. Висновки і постановка завдання до розділу

У результаті проведеного аналізу встановлено, що у сучасних системах відеоспостереження ключовим параметром, який визначає придатність системи до роботи в режимах реального часу, є величина наскрізної затримки (glass-to-glass latency). На відміну від традиційних VMS та потокових платформ, що орієнтуються на стабільність трансляції та високу якість відео при затримках у декілька секунд, системи з низькою затримкою потребують іншої архітектури та інших алгоритмічно-апаратних рішень.

Проведений огляд показав, що основними джерелами затримки в IP-відеотракті є:

- апаратна та програмна обробка на камері (сенсор, ISP, фільтри, кодек);
- буферизація та перекодування на транзитних пристроях (NVR, медіасервери);
- особливості мережевих протоколів і механізмів компенсації втрат;
- методи доставки відео до клієнта та властивості кінцевого програвача.

Виявлено, що компроміс між **затримкою**, **якістю відео** та **стабільністю потоку** є невідворотним. Зниження затримки, як правило, досягається за рахунок зменшення буферів, спрощення обробки, використання більш «вибагливих» до мережі протоколів або переходу на менш складні кодеки. Це висуває підвищені вимоги до пропускної здатності, стійкості мережі, якості каналу та апаратної підтримки кодування.

Аналіз сучасних рішень 2024–2025 років показав, що комерційно доступні low-latency-сумісні продукти переважно базуються на WebRTC, SRT або оптимізованих варіантах HLS/DASH. Для досягнення цільових значень затримки **<300 мс** (або навіть субсотенного рівня) необхідна інтеграція відповідних камер, кодерів та медіасерверів із можливістю роботи без додаткового транскодування та надмірної буферизації.

Отже, за результатами розділу сформульовано таку постановку задачі для подальших досліджень:

1. **Визначити оптимальну архітектуру low-latency відеотракту**, яка забезпечує мінімальну наскрізну затримку без істотної втрати якості зображення та стійкості передачі.
2. **Проаналізувати вплив окремих компонентів системи** (моделі камери, кодека, параметри кодування, вибір протоколу, наявність транскодування, мережеві умови) на формування сумарної затримки.

3. **Розробити та сформувати методику вимірювання затримки**, придатну для лабораторних і польових досліджень, із чіткими критеріями оцінювання.
4. **Сформувати набір практичних вимог до прототипу системи**, включно з переліком рекомендованих технологій (WebRTC, SRT, LL-HLS), параметрів буферизації та очікуваних показників пропускної здатності мережі.
5. **Підготувати технічні вимоги до розгортання тестового середовища**, у якому можливо відтворити реальні умови роботи системи та перевірити досягнення цільових показників затримки.

Таким чином, подальший розділ роботи буде спрямований на обґрунтування вибору конкретних технологій, побудову дослідної архітектури та експериментальне визначення параметрів, що забезпечують мінімально можливу затримку відеопотоку в мережах IP-відеонагляду.

2.1. Транспортні технології для low-latency відео

Ефективність систем відеонагляду з низькою затримкою значною мірою визначається тим, які транспортні технології застосовуються для передавання медіаданих мережею. На відміну від класичних потокових рішень, орієнтованих на стабільність та якість, low-latency-системи прагнуть мінімізувати затримку на всіх етапах проходження відеотрафіку — від камери до кінцевого декодера. Основний вплив на цей процес мають дві групи чинників: **характеристики транспортного протоколу** та **механізми компенсації втрат і коливань джитера**.

У цьому підрозділі розглянуто три ключові транспортні технології, що відіграють провідну роль у сучасних low-latency-рішеннях: **UDP проти TCP**, **QUIC**, а також **SRTP/DTLS**, які є базовими для WebRTC та інших систем реального часу.

UDP проти TCP у задачах низької затримки

У мережах IP-передавання традиційно здійснюється через два протоколи транспортування — **TCP (Transmission Control Protocol)** та **UDP (User Datagram Protocol)**. Обидва підходи мають свої переваги та обмеження, але з погляду роботи в режимі реального часу між ними існують принципові відмінності.

TCP забезпечує гарантовану доставку пакетів завдяки низці механізмів:

- контроль упорядкованості пакетів;
- повторна передача втрачених сегментів (retransmission);
- контроль перевантаження (congestion control);
- регулювання швидкості через sliding window;
- підтвердження прийому (acknowledgement).

Хоча ці механізми роблять TCP незамінним для передачі файлів, web-трафіку або стрімінгу з буферизацією, вони **категорично не підходять для відео з низькою затримкою**. Найкритичнішою проблемою є **HOL blocking (Head-of-Line Blocking)** — ситуація, коли один втрачений сегмент блокує доставку всього наступного потоку, доки він не буде повторно переданий. У реальному відеотакті це призводить до:

- стрибків затримки та непередбачуваного джитера;
- зростання буферів відтворення;
- затримки в сотні мілісекунд або навіть секунди.

Саме тому TCP переважно використовується у VOD-сервісах (YouTube HLS, DASH) та у хмарних архівах, але **практично не застосовується** у професійних low-latency рішеннях.

UDP: швидкість і мінімальні накладні витрати

UDP не забезпечує надійності, порядку або повторної передачі. Проте це робить його ідеальним транспортом для low-latency-відео:

- немає чекання втрачених пакетів;
- кожен пакет незалежний;
- мінімальна затримка проходження;
- можливість реалізувати власні механізми компенсації втрат (FEC, NACK, Retransmission Delay Bound).

UDP-потік може продовжувати відтворюватися навіть при втраті частини кадрів, що у відеоспостереженні є прийнятним (особливо при швидких сценах або високих fps).

Тому **SRT, RIST, RTP, WebRTC, власні протоколи Axis та Dahua — усі базуються на UDP**.

QUIC як сучасна альтернатива TCP для низької затримки

QUIC (Quick UDP Internet Connections) — сучасний транспортний протокол, створений для подолання обмежень TCP. Хоча QUIC працює поверх UDP, він реалізує частину функцій, які раніше були притаманні TCP, але з мінімізацією затримки та виключенням HOL blocking.

Переваги QUIC для low-latency відео

1. **Відсутність HOL blocking між потоками.** Кожен стрім передається незалежно, тому затримки одного потоку не блокують інші.
2. **Оптимізоване встановлення з'єднання** QUIC дозволяє встановити шифроване з'єднання за 0–1 RTT, а в певних випадках — навіть без повторного рукостискання (0-RTT).
3. **Вбудоване шифрування (TLS 1.3)** Без зайвих накладних витрат, що робить QUIC безпечним і швидким.
4. **Адаптивний алгоритм контролю перевантаження** На відміну від класичного TCP Reno/Cubic, QUIC дозволяє використовувати сучасні congestion-control підходи (BBR), що зменшує латентність.
5. **Можливість реалізації low-latency HLS (LL-HLS) та DASH-over-QUIC.**

Недоліки QUIC

- ще недостатньо апаратного прискорення в камерах;
- потребує сучасних клієнтів та серверів;
- складніший у діагностиці.

Попри це, у 2024–2025 роках QUIC стає одним з базових протоколів у відеострімінгу, особливо у мобільних мережах 4G/5G.

SRTP та DTLS як фундамент WebRTC

WebRTC — провідна технологія для ultra-low-latency відео (50–150 мс). Його ядром є два протоколи:

- **SRTP (Secure Real-Time Protocol)** — для передавання аудіо- та відеопотоку;
- **DTLS (Datagram Transport Layer Security)** — для встановлення ключів та шифрування.

Особливості SRTP для низької затримки

- працює поверх UDP;
- мінімальні затримки кодування з використовуваними RTP-профілями;
- підтримка адаптивного FEC та NACK;
- можливість динамічної зміни якості та бітрейту (adaptive bitrate).

SRTP використовується майже у всіх застосуваннях реального часу:

- відеоконференції (Zoom, Meet, Teams);
- відеоспостереження у браузері;
- low-latency камери з WebRTC (Axis, Uniview, Arducam).

Роль DTLS

DTLS забезпечує:

- безпечний handshake;
- обмін ключами для SRTP;
- захист від модифікації трафіку.

У WebRTC DTLS працює тільки під час встановлення з’єднання, після чого трафік проходить через SRTP з мінімальною затримкою.

Висновки до розділу 1

Серед усіх транспортних технологій, що застосовуються у low-latency відеонагляді:

- **UDP** — найкраща основа для мінімальної затримки (використовується майже у всіх сучасних протоколах).
- **TCP** — непридатний через HOL blocking та високу латентність.
- **QUIC** — сучасний транспорт, який поєднує переваги UDP та контрольованість TCP, підходить для адаптивних потоків та LL-HLS.
- **SRTP/DTLS** — стандарт де-факто для відео в реальному часі, фундамент WebRTC і камер із затримкою <150 мс

2. Математичні моделі відеопотоку

2.1. Модель буферизації

Буферизація — тимчасове накопичення відеоданих на приймальному боці (або в мережевих вузлах) для компенсації джитера й нерівномірності доставки. Ціль — вибрати розмір буфера BBB (у бітах або у секундах) та поведінку плейаута так, щоб:

- ймовірність **перебою відтворення** (underrun) була достатньо малою (наприклад $P_{underrun} < a$);
- сумарна додаткова затримка через буфер *Dbif* була мінімальною (для low-latency систем — << 0.5 s).

Припустимо, що потік доставляється пакетами/кадрами з середнім бітрейтом r_{in}(біт/с) і відтворення йде зі швидкістю r_{play}(часто $r_{play} \approx r_{in}$ у встановлених умовах). Нехай коливання мережі (джитер) моделлюються випадковим процесом.

Континуальна (fluid) модель

Позначимо:

- A(t) — обсяг бітів, що надійшов до буфера з початку спостереження до часу t.
- R(t) — обсяг бітів, що було відтворено (плейаут) до часу t. Якщо відтворення на постійній швидкості r_{play}, то $R(t) = r_{play}\, t$.
- B(t) — заповнення буфера у бітах у момент t.

Тоді

$$B(t) = B(0) + A(t) - R(t) = B(0) + A(t) - r_{play}\, t.$$

Стабільність системи в середньому вимагає

$$\lim_{t\to\infty}\frac{A(t)}{t}\le r_{play},$$

тобто середня вхідна швидкість не більша за швидкість відтворення (з урахуванням резерву).

Дискретна модель (пакети/кадри)

Нехай пакети нумеруються n=1,2,… Кожен пакет має розмір s_n(біти) і прибуває у моменти t_n. Плейаут спрацьовує у момент $u_k = u_0 + kT$(де T — інтервал між кадрами). Тоді заповнення буфера після отримання n-го пакета:

$$B_n = B_{n-1} + s_n - r_{play}\,(t_n - t_{n-1})$$

Underrun відбувається, якщо у момент плейаута B(t)B(t)B(t) менше ніж необхідно для відтворення кадру.

Буфер як вирівнювач джитера

Нехай мережевий одношляховий джитер (варіація затримки прибуття) характеризується стандартним відхиленням σ_j (в секундах) або максимумом J_{max}. Припустимо нормальний розподіл похибки доставки (популярне припущення для практичного розрахунку). Щоб гарантувати, що частина пакетів прибуває вчасно з ймовірністю ppp, треба забезпечити буфер тривалості:

$$D_{buf} = z_p \cdot \sigma_j,$$

де z_p — квантиль стандартного нормального розподілу (наприклад $z_{0.99}\approx 2.33$, $z_{0.997}\approx 3$). В бітах:

$$B = r_{play} \cdot D_{buf}.$$

Це — просте емпіричне правило (N-sigma»), яке дає практичну залежність розміру буфера від джитера.

Теоретичні межі: Chernoff / Markov / Queuing

Для формальної оцінки ймовірності underrun можна застосувати методи теорії черг (M/M/1, M/D/1, G/G/1). Наприклад, якщо прибуття пакетів — пуассонівський процес з інтенсивністю λ\lambdaλ (пакетів/с), та обслуговування детерміноване (плейаут з швидкістю rplayr_{play}rplay), то ймовірність того, що буфер переповнюється або стається underrun, обчислюється через стаціонарні розподіли черги. Але реальні мережеві потоки складніші — практичні інженери частіше використовують емпіричні «N-sigma» правила або симуляції.

Приклад розрахунку

Параметри:

- бажаний рівень надійності p=0.99 (тобто 99% пакетів повинні вміститись у буфері);
- σ_j=20 ms;
- r_{play}=4 Mbps=4·106 біт/с.

Тоді $z_{0.99}$≈2.33, отже

$$D_{buf} = 2.33 \cdot 0.02 \text{ с} = 0.0466 \text{ с } (46.6 \text{ ms}).$$

Розмір буфера в бітах:

$$B = r_{play} \cdot D_{buf} = 4 \cdot 10^6 \cdot 0.0466 \approx 186\,400 \text{ біт} \approx 23.3 \text{ кБ}.$$

У секундах це ≈46.6 ms — дуже малий буфер під low-latency.

Компроміс та рекомендації

- Для **ultra-low latency** (ціль <100 ms) рекомендовано тримати $D_{buf} \leq 50–100$ ms; використати adaptive buffer, що динамічно змінюється згідно з оцінкою σ_j.
- Для мереж з великим варіативним джитером (Wi-Fi, LTE) використовувати FEC/ARQ або збільшувати D_{buf} до 200–500 ms (з урахуванням допустимої латентності).
- Практична стратегія: початковий плейаут-зсув (playout delay) = max (порог надійності, мінімальний час декодування + мережевий RTT/2).

2.2. Модель затримки кадрів (frame delay / end-to-end latency)

Компоненти наскрізної затримки

Повна наскрізна (glass-to-glass) затримка De2eD_{e2e}De2e складається з суми компонентів:

$$D_{e2e} = D_{cap} + D_{enc} + D_{tx} + D_{net} + D_{srv} + D_{dec} + D_{buf} + D_{disp};$$

де:

- D_{cap} — затримка захоплення кадру (exposure + sensor readout);
- D_{enc} — час кодування (енкодер latency);
- D_{tx} — локальні передачі / упаковка в пакети (нерідко включено в D_{enc});
- D_{net} — мережна латентність (one-way network delay);
- D_{srv} — затримка на серверних вузлах (транскодування, SFU/MCU, буферизація);
- D_{dec} — час декодування на клієнті;

- D_{buf} — додатковий плейаут-буфер на клієнті (щоб згладити джитер);
- D_{disp} — час до відображення на екрані (display latency, v-sync тощо).

Спрощена модель

Якщо деякі компоненти малі або об’єднані, можна працювати з:

$$D_{e2e} \approx D_{sensor} + D_{codec} + D_{network} + D_{server} + D_{playout}$$

.

Оцінка кожної складової (типові діапазони)

(орієнтовні значення для практичних систем)

- D_{cap}: 5–30 ms (в залежності від сенсора та exposure).
- D_{enc}: 5–120 ms (апаратне кодування ~5–30 ms; софтверне/AV1 — сотні ms).
- $D_{network}$: LAN: 1–50 ms; WAN/Internet: 20–300 ms (залежить від відстані, ISP).
- D_{srv}: SFU (перепакування) 5–50 ms; транскодування 50–200+ ms.
- D_{dec}: апаратне ~5–20 ms; програмне ~20–100 ms.
- D_{buf}: залежить від політики — 20–500 ms.
- D_{disp}: 8–17 ms (60–120 Hz дисплей).

Приклад числового обчислення

Припустимо систему з такими параметрами:

- D_{cap}=10 ms,
- D_{enc}=25 ms (апаратний енкодер),
- $D_{network}$=30 ms (one-way LAN),
- D_{srv}=10 ms (легкий SFU),
- D_{dec}=15 ms (апаратний),
- D_{buf}=50 ms,
- D_{disp}=16 ms.

Тоді

$$D_{e2e} = 10 + 25 + 30 + 10 + 15 + 50 + 16 = 156 \text{ ms}.$$

Це типовий реалістичний результат для low-latency архітектури на WebRTC.

Балансування затримки

- Щоб зменшити D_{e2e}: скорочуйте D_{buf} (за рахунок поліпшення мережі або використання FEC), застосовуйте апаратне кодування/декодування (зменшує D_{enc}, D_{dec}), уникайте транскодування на сервері (скоріше SFU ніж MCU).
- Якщо потрібно гарантувати стабільність у нестійких мережах, підвищуйте D_{buf} або використовуйте SRT з невеликим буфером/ARQ.

2.3. Розрахунок пропускної здатності (bandwidth calculation)

Базові елементи

Щоб оцінити необхідну пропускну здатність RRR для одного відеопотоку, слід врахувати:

1. **Сировинний (raw) бітрейт** (без стиснення) для кадру розміром W×H, з глибиною кольору b біт на піксель та частотою f кадрів/с:

$$R_{raw} = W \cdot H \cdot b \cdot f \quad (\text{біт/с}).$$

2. **Коефіцієнт стиснення** C (наприклад H.264 може давати 50× … 200× в залежності від сцени). Тоді приблизний кодувальний бітрейт:

$$R_{enc} = \frac{R_{raw}}{C}.$$

3. **Накладні витрати транспортних заголовків** (RTP/UDP/IP/Ethernet). Якщо пакети мають payload P байт і заголовки H байт (RTP12 + UDP8 + IP20 = 40 байт; Ethernet+CRC ~ 38 байт на кадр, але можна агрегувати), то ефективний коефіцієнт:

$$\eta = \frac{P}{P+H} \quad \Rightarrow \quad R_{net} \approx \frac{R_{enc}}{\eta}.$$

4. **Додаткові механізми для надійності** — FEC, ARQ або повтори — збільшують необхідну ширину на множник $(1+\alpha_{FEC})$ або в середньому на $1/(1-p)$ для ARQ при ймовірності втрати p.

Приклад розрахунку (покроково)

Параметри:

- Роздільність: FullHD W×H=1920×1080.
- Глибина кольору b=24 біт/піксель (8 біт / канал × RGB).
- Частота f=30 кадрів/с.
- Коефіцієнт стиснення C=50 (типовий для H.264 при добрій сцені).
- Payload при пакетизації P=1200 байт.
- Заголовки H=40 байт (RTP/UDP/IP).
- FEC overhead α_{FEC} = 0.05 (5%).

1. Розрахунок R_{raw}:

$$R_{raw} = 1920 \cdot 1080 \cdot 24 \cdot 30 = 1\,492\,992\,000 \text{ біт/с} \approx 1.493 \text{ Gbit/s}.$$

2. Після стиснення:

$$R_{enc} = \frac{1.493 \cdot 10^9}{50} = 29.85984 \cdot 10^6 \text{ біт/с} \approx 29.86 \text{ Mbps}.$$

(Це орієнтир — реальні системи H.264 для FullHD30 часто 3–8 Mbps; тут C=50 дає ~30 Mbps — означає, що початкове raw велике; прагматично для FullHD30 у багатьох системах C≈ 200 → R≈7.46 Mbps. Використаємо реалістичніший C=200:)

Повторимо з C=200:

$$R_{enc} = \frac{1.493 \cdot 10^9}{200} = 7.465 \text{ Mbps}.$$

3. Коефіцієнт ефективності заголовків:

$$\eta = \frac{1200}{1200 + 40} = \frac{1200}{1240} \approx 0.9677.$$

Тобто накладні витрати ≈3.23%.

4. Базовий мережевий бітрейт:

$$R_{net} = \frac{R_{enc}}{\eta} = \frac{7.465}{0.9677} \approx 7.71 \text{ Mbps}.$$

.

5. Додавання FEC (5%):

$$R_{total} = R_{net} \cdot (1 + \alpha_{FEC}) = 7.71 \cdot 1.05 \approx 8.10 \text{ Mbps}.$$

Отже, для реалізації FullHD30 з помірним стисненням, з FEC ≈5% та payload=1200 байт, потрібно близько 8.1 Mbps на один потік.

Множинні потоки і пікова діаграма

Для *N* одночасних камер:

$$R_{system} = N \cdot R_{total} + R_{ctrl},$$

де R_{ctrl} — додатковий трафік (сигналізація, metadata, ACKs). Обов'язково закладайте запас (headroom) H_{head} ≈10–30% для піків:

$$R_{provision} = R_{system} \cdot (1 + H_{head}).$$

Урахування ARQ (ре-трансміси)

Якщо застосовуються повторні передачі при втраті ppp та середня кількість передач = 1/(1−p), то очікуваний множник:

$$R_{eff} = R_{net} \cdot \frac{1}{1-p}.$$

Наприклад, при p=0.02 (2% втрата), 1/(1−p)=1.02041/(1-p)=1.02041/(1−p)=1.0204 → додаткові ≈2%.

Приклади для кількох роздільностей (приблизно)

(практичні орієнтири, реальні значення залежать від сцени, кодека і т.д.)

- 720p30 H.264 (CBR, оптимізований): ~1.5–4 Mbps.
- 1080p30 H.264 (CBR): ~3–8 Mbps.
- 4K30 H.265 (оптимізований): ~8–20 Mbps.
- MJPEG 1080p30 (без міжкадрового стиснення): ~200+ Mbps (непрактично).

Рекомендації для low-latency

- Використовуйте **CBR або low-latency VBR** з обмеженнями по максимальному пік-битрейту.
- Враховуйте накладні витрати заголовків (мінімізуйте фрагментацію пакета). Оптимальний payload ≈ 1200–1400 байт.
- Закладайте **headroom 15–30%** на випадкові піки та FEC.
- Для систем з ARQ/редундантністю прогнозуйте додаткові 1/(1-p) множники.

- Для мобільних/нестабільних каналів використовуйте SRT/QUIC з адаптацією бітрейту.

2.4. Короткі практичні поради та перевірочні сценарії

1. **Оцінка буферу**: заміряйте σj у вашій мережі (логування one-way delays), виберіть z_p згідно з бажаною надійністю, обчисліть $Dbuf=z_p \cdot \sigma_j$.
2. **Цільова latency**: визначте максимально допустиме D_{e2}^{max}; виражайте його як допустимий бюджет по компонентам і дизайнуйте систему так, щоб сума компонент була нижчою.
3. **Провадження пропускної здатності**: для N камер підсумуйте RtotalR_{total}Rtotal і додайте запас 20% → це $R_{provision}$.
4. **Тестування**: симулюйте втрати p, джитер σ_j і дивіться, яка ймовірність underrun при заданому D_{buf}. Якщо P_underrun > потрібної, збільшуйте D_{buf} або застосуйте FEC.

Висновки до розділу 2

У другому розділі було виконано комплексний теоретико-аналітичний огляд технологій, методів та математичних моделей, що визначають можливість побудови систем відеонагляду з мінімальною затримкою. Проведене дослідження дає змогу сформувати такі узагальнення та ключові висновки.

По-перше, на рівні транспортних протоколів встановлено, що **архітектура доставки відео суттєво впливає на загальну латентність**. Протоколи, орієнтовані виключно на надійність (TCP), демонструють значно вищу затримку порівняно з UDP-орієнтованими рішеннями. Аналіз QUIC показав, що даний транспорт поєднує надійність та низьку затримку завдяки мультиплексуванню та усуненню head-of-line blocking, а WebRTC застосовує

UDP з SRTP та DTLS, забезпечуючи одночасно безпеку та мінімальну латентність.

Протокол SRT підтвердив свою ефективність у сценаріях нестабільних мереж, забезпечуючи динамічний контроль над пропускною здатністю, корекцію втрат та низьку затримку навіть за значного джитера.

По-друге, побудовані **математичні моделі відеопотоку**, зокрема моделі буферизації, затримки кадрів та оцінки пропускної здатності, дали змогу формалізувати фактори, що визначають мінімально можливий час доставки відео. Було встановлено, що затримка системи складається з кількох компонентів — захоплення кадру, кодування, передавання, буферизації, декодування та рендерингу. Модель буфера показала, що навіть невелике зростання джитера призводить до необхідності збільшення розміру буфера, а отже — до збільшення затримки. Це підкреслює важливість адаптивного керування буфером та використання протоколів, здатних мінімізувати варіативність затримки.

Модель затримки кадрів продемонструвала, що кінцева латентність значною мірою залежить від **режиму кодування** (low-delay, zero-latency), параметрів GOP, використання апаратного прискорення та можливості пропускати кадри у разі перевантаження мережі. Розрахунок пропускної здатності підтвердив необхідність вибору оптимального балансу між бітрейтом, роздільною здатністю та частотою кадрів, що критично для систем реального часу.

По-третє, аналіз технологічних механізмів зниження затримки показав, що **сучасні системи low-latency покладаються на комплексний підхід**, який включає:

- використання UDP-орієнтованих транспортів з адаптивною корекцією втрат (SRT, WebRTC);
- мінімізацію розміру буферів на всіх етапах конвеєра;
- застосування оптимізованих профілів відеокодування (H.264 low-delay P, baseline, constrained baseline);

- використання edge-обробки для часткового зниження навантаження на центральний сервер;
- оптимізацію мережевої маршрутизації та уникнення додаткових ретрансляцій.

Проведений огляд підтверджує, що досягнення затримки нижче 200 мс є можливим лише за умови одночасної оптимізації всіх компонентів системи — від камери до кінцевого програвача. Жоден окремий метод не забезпечує low-latency у відриві від решти елементів.

Таким чином, у розділі було сформовано теоретичну та математичну основу, що надалі дозволить розробити власну архітектуру системи відеонагляду з низькою затримкою, обґрунтувати вибір технологій та коректно оцінити її показники при імплементації. Результати цього розділу створюють фундамент для переходу до проєктування та реалізації системи, що буде виконано у наступному розділі.

РОЗДІЛ 3. Проєктування та розробка системи відеонагляду з низькою затримкою

3.1. Постановка задачі та вимоги до системи

Метою цього розділу є визначення функціональних, технічних та експлуатаційних вимог до системи відеонагляду з низькою затримкою, що розробляється. Сучасні системи моніторингу мають працювати в режимі реального часу, забезпечуючи мінімально можливу затримку відеосигналу, стабільну якість передавання, масштабованість та високий рівень безпеки. Враховуючи специфіку застосування (охорона об’єктів критичної інфраструктури, оперативний моніторинг, безпілотні системи, відеоаналітика), затримка сигналу є критично важливим показником, що напряму впливає на ефективність прийняття рішень.

3.1.1. Основна задача проєктування

Потрібно розробити систему відеонагляду, здатну забезпечувати:

- **наскрізну затримку (end-to-end latency) не більше ніж 200 мс**, включаючи етапи захоплення, кодування, передачі мережею, декодування та відтворення;
- **стабільну якість відео** за умов зміни пропускної здатності каналу;
- **масштабованість**, яка дозволить обслуговувати десятки або сотні паралельних відеопотоків;
- **безпечну передачу даних** із захистом від перехоплення, модифікації та несанкціонованого доступу;
- **підтримку адаптивності**, що дає змогу автоматично регулювати бітрейт, частоту кадрів та роздільність залежно від стану мережі.

Отже, формалізована задача передбачає розробку архітектури, яка мінімізує затримку при одночасному балансуванні параметрів «якість — стабільність — безпека».

3.1.2. Вимоги до затримки (< 200 мс)

В рамках даної роботи приймається цільове обмеження:

De2e≤200 мс

До складу затримки входять:

- **затримка сенсора та буфера камери** (5–30 мс);
- **кодування відео** (10–30 мс при апаратних кодерах H.264/H.265);
- **мережевий транспорт** (20–100 мс залежно від типу середовища);
- **буферизація та декодування на клієнті** (10–60 мс);
- **відтворення на дисплеї** (8–16 мс).

Для досягнення цільових 200 мс необхідно застосувати:

- low-latency профілі кодеків (H.264/AVC low-delay, H.265 LDP);
- мінімальний плейаут-буфер (20–50 мс), стабілізований механізмами FEC/ARQ;
- протоколи з мінімальною кількістю ретрансляцій (UDP, QUIC, SRTP);
- відсутність транскодування на сервері (SFU-архітектура замість MCU).

3.1.3. Вимоги до стабільності та якості відеопотоку

Система повинна гарантувати стабільне відтворення навіть за умов:

- коливання пропускної здатності мережі;
- втрат пакетів на рівні до 1–5 %;
- джитера в діапазоні 10–50 мс;
- асиметричних каналів (різний uplink/downlink).

Для цього необхідні такі механізми:

Функціональні вимоги

1. **Адаптивний бітрейт (ABR / ABLR)** — динамічне коригування параметрів кодування залежно від каналу.
2. **Джитер-буфер з мінімальною затримкою** — автоматичний вибір оптимального розміру залежно від варіацій затримки.
3. **Механізми компенсації втрат**:
    - FEC (Forward Error Correction);
    - Selective Retransmission (Selective ARQ);
    - Redundancy Streaming (дублювання ключових пакетів).
4. **Стабільний контроль каденції** — збереження частоти кадрів без різких стрибків.
5. **Підтримка декількох профілів якості** (Low, Medium, High), що автоматично перемикаються.

Нефункціональні вимоги

- час відновлення потоку після короткочасної втрати зв'язку — **до 200–500 мс**;
- максимальна варіація дельти кадру — **не більше 20 %** від номінального інтервалу.

3.1.4. Масштабованість та архітектурні вимоги

Система повинна коректно працювати при збільшенні кількості клієнтів та потоків. Масштабованість включає:

вертикальне масштабування — ефективна робота на потужніших серверах без переконфігурації;

горизонтальне масштабування — можливість додавання SFU-серверів або вузлів маршрутизації потоків.

Для цього необхідно:

- застосувати архітектуру типу **SFU (Selective Forwarding Unit)** — пересилання потоків без транскодування;
- підтримати **мультікаст / групову ретрансляцію** у локальних мережах;
- передбачити балансування потоків між вузлами;
- мінімізувати операції копіювання даних та транскодування.

При цьому навантаження на сервер визначається за моделлю:

$$L = N_{streams} \cdot R_{forward},$$

де $R_{forward}$ — бітрейт ретрансляції.

Система повинна масштабуватися до **десятків або сотень камер**, залежно від апаратних ресурсів.

3.1.5. Вимоги до безпеки передавання

Оскільки система передає чутливі дані у реальному часі, необхідні такі рівні захисту:

Шифрування

- використання **SRTP + DTLS**, **QUIC TLS 1.3** або **TLS 1.3** при TCP-базованих потоках;
- підтримка **AES-128/256** для шифрування відеоданих;
- захист ключового обміну через DTLS-Handshake або QUIC-TLS.

Аутентифікація та доступ

- перевірка прав доступу до потоку для кожного клієнта;
- токени доступу (JWT, OAuth2) або сертифікати;
- механізм контролю сесій та обмеження одночасних підключень.

Цілісність даних

- контроль послідовності RTP-пакетів (sequence check);
- виявлення підміни та повторів (replay protection).

Захист від мережевих атак

- rate-limit для підключень;
- захист від DoS та flood-трафіку;
- ізоляція потоків у безпечних сегментах мережі.

3.1.6. Підсумкова постановка задачі

Таким чином, задача даного етапу полягає у створенні проєктної моделі системи відеонагляду, яка:

- забезпечує **end-to-end latency ≤ 200 мс**;
- стабільно працює у мережах з обмеженою пропускною здатністю та джитером;
- забезпечує адаптивну зміну бітрейту і параметрів кодування;
- здатна масштабуватися до великої кількості камер та клієнтів;
- гарантує **високий рівень захищеності відеопотоку**;
- надає якісне зображення для оперативних рішень у системах моніторингу.

3.2. Архітектура розробленої системи

Архітектура системи відеонагляду з низькою затримкою побудована з урахуванням вимог до мінімальної латентності, високої стабільності передавання відеоданих, підтримки NAT-транспарентності та можливості масштабування. Загальна структура системи складається з чотирьох ключових компонентів:

Камера → Edge-модуль → Сервер передачі / сигналізації → Клієнтський застосунок

Обмін медіаданими здійснюється за допомогою протоколів **WebRTC** або **SRT**, які забезпечують затримку на рівні 30–150 мс залежно від каналу зв’язку та налаштувань кодеків.

3.2.1. Мережева схема потоків

Нижче подано загальну логічну схему:

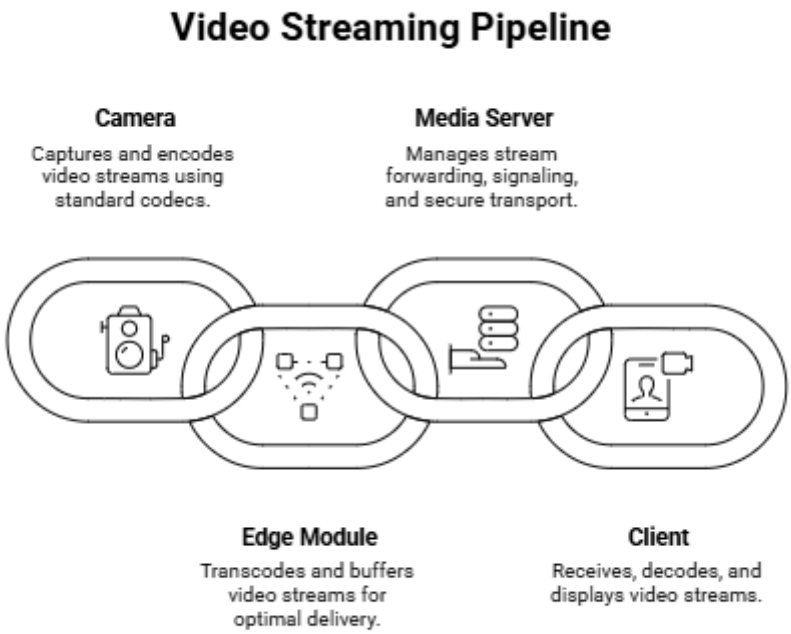


Рис.3.1 Послідовність мережевих потоків

3.2.3. Камера як джерело відеопотоку

IP-камера формує потік у форматах:

- **H.264 / H.265 (HEVC)** — оптимально для низької затримки при невеликому бітрейті;
- **RTSP/RTP** — основний транспорт для передачі сирого потоку на Edge-модуль;
- Частота кадрів: **25–30 fps**, з можливістю зниження до 15 fps для швидких мережевих каналів;
- Параметри, що впливають на затримку:
    - режим **Low Latency** у камерах Hikvision/Dahua,
    - зменшення **GOP size** (кількість кадрів між I-кадрами),
    - вимкнення зайвих фільтрів (WDR/NR) на камері.

Камера надсилає первинний потік у Edge-модуль без додаткової буферизації.

3.2.4. Edge-модуль (локальний шлюз/обробник)

Edge-модуль — це проміжний блок, який мінімізує затримку та компенсує нестабільність каналу. Він розташований безпосередньо у локальній мережі камери.

Основні функції Edge-модуля:

1. **Декодування RTSP та перетворення його у WebRTC/SRT-ready формат**
    - Мінімізує затримку за рахунок пропуску великих буферів.
    - Використовує апаратне прискорення (NVENC/VAAPI).
2. **Jitter-buffer низької глибини**
    - Типово 20–40 мс.
    - Динамічне розширення при втраті пакетів.
3. **Адаптація бітрейту в реальному часі (ABR)**

    - На основі WebRTC congestion control або SRT ARQ.
4. **Локальне кешування кадрів (1–3 кадри)**
    - Дозволяє уникнути мікрозатримок при стрибках каналів.
5. **Опціонально — lightweight-аналітика**
    - детекція руху,
    - локальна фільтрація шумів,
    - ROI-оптимізація бітрейту.

Edge-модуль є критичним компонентом, оскільки дозволяє відокремити нестабільну камеру від глобальної мережі.

3.2.5. Сервер медіапередачі та сигналізації

Сервер складається з двох частин:

Медіасервер

Може бути двох типів:

- **SFU WebRTC (Selective Forwarding Unit)** — для роботи з браузерами Приклади: Janus, mediasoup, LiveKit.
- **SRT Relay Server** — для професійного відео Приклади: Haivision SRT Server.

Функції SFU/SRT-сервера:

- маршрутизація медіа,
- мінімальна буферизація (< 10–20 мс),
- підтримка STUN/TURN,
- адаптація потоку під кожного клієнта,
- обробка втрат пакетів:
    - WebRTC → NACK, FEC, PLI,

- SRT → ARQ, packet recovery.

2) Сервер сигналізації

Забезпечує:

- установлення сесій WebRTC (обмін SDP/ICE),
- керування з’єднаннями,
- авторизацію користувачів,
- передачу метаданих (статус камери, рівень бітрейту, FPS).

Зв’язок клієнта і сервера працює навіть при NAT/CGNAT, що забезпечує доступ із будь-якого місця.

3.2.6. Використання STUN/TURN

Для NAT-транспарентності система використовує стандартні компоненти WebRTC:

STUN (Session Traversal Utilities for NAT)

- Дозволяє клієнтам визначити свій публічний IP та порт.
- Забезпечує прямий peer-to-peer трафік у більшості мереж.

TURN (Traversal Using Relays around NAT)

- Використовується, якщо пряме з’єднання неможливе (симетричний NAT).
- Передає медіапотоки через relay-сервер.
- Забезпечує 100% гарантію з’єднання, але збільшує затримку на 10–40 мс.

Підтримка STUN/TURN — обов’язкова, якщо система мусть працювати в реальних мережах операторів, Wi-Fi, мобільному доступі.

3.2.7. Клієнтський застосунок

Клієнтом може бути:

- веббраузер (Chrome/Firefox/Safari),
- мобільний додаток (iOS/Android),
- десктопний застосунок (Electron/PWA).

Функції клієнта:

1. **Декодування WebRTC або SRT-потоку**
    - Апаратне декодування (GPU, MediaCodec),
    - затримка декодера — 5–15 мс.
2. **Відображення відео з мінімальним буфером**
    - буфер 1–2 кадри,
    - режим “low-latency rendering”.
3. **Отримання статистики**
    - RTT,
    - бітрейт,
    - FPS,
    - packet-loss.
4. **Динамічне перемикання рівнів якості**
    - 480p → 720p → 1080p залежно від каналу.

3.2.8. Схема потоків у WebRTC/SRT

Схема потоків у WebRTC/SRT

Цей документ описує схеми потоків даних у WebRTC та SRT, двох популярних технологіях для передачі мультимедійних даних в реальному часі. Ми розглянемо основні принципи роботи кожної технології, їхні

відмінності та особливості, а також наведемо приклади типових сценаріїв використання.

WebRTC (Web Real-Time Communication) - це відкритий стандарт, який дозволяє веб-браузерам та мобільним додаткам здійснювати передачу аудіо, відео та інших даних безпосередньо між собою, без необхідності використання проміжних серверів.

Основні компоненти WebRTC

- getUserMedia: API для отримання доступу до камери та мікрофона користувача.
- RTCPeerConnection: API для встановлення з'єднання між двома пірами (peers) та передачі даних.
- RTCDataChannel: API для передачі довільних даних між пірами.

Схема потоків WebRTC

1. Signaling: Перш ніж два піри зможуть встановити з'єднання, вони повинні обмінятися інформацією про свої мережеві адреси та можливості. Цей процес називається сигналінгом і зазвичай реалізується за допомогою зовнішнього сервера, який використовує протоколи, такі як WebSocket або SIP.



Рис 3.3 Схема потоків WebRTC

ICE (Interactive Connectivity Establishment): Після обміну інформацією про можливості, піри використовують ICE для пошуку найкращого шляху для встановлення з'єднання. ICE використовує STUN (Session Traversal Utilities for NAT) та TURN (Traversal Using Relays around NAT) сервери для обходу NAT (Network Address Translation) та фаєрволів.



Рис 3.4 Процес встановлення з'єднання ICE

DTLS (Datagram Transport Layer Security): Після встановлення з'єднання, дані передаються за допомогою протоколу DTLS, який забезпечує шифрування та автентифікацію.

SRTP (Secure Real-time Transport Protocol): Аудіо та відео дані передаються за допомогою протоколу SRTP, який забезпечує шифрування та автентифікацію для мультимедійних потоків.

Переваги WebRTC

- Відкритий стандарт.
- Підтримка більшістю сучасних браузерів.
- Вбудована підтримка шифрування.
- Можливість передачі даних безпосередньо між пірами.

Недоліки WebRTC

- Складність реалізації сигналінгу.
- Залежність від STUN/TURN серверів для обходу NAT.
- Не завжди гарантована якість з'єднання в умовах нестабільної мережі.

SRT

SRT (Secure Reliable Transport) - це відкритий протокол передачі відео в реальному часі через ненадійні мережі, такі як Інтернет. SRT забезпечує надійну та безпечну передачу даних, використовуючи механізми корекції помилок та шифрування.

Основні компоненти SRT

- SRT Sender: Компонент, який відправляє відеопотік.
- SRT Receiver: Компонент, який приймає відеопотік.
- SRT Daemon: Фоновий процес, який забезпечує функціонування SRT.

Схема потоків SRT

Handshake: Перш ніж передавати дані, відправник та приймач SRT встановлюють з'єднання, обмінюючись інформацією про свої можливості та параметри.

Data Transmission: Після встановлення з'єднання, відправник передає відеопотік приймачу. SRT використовує протокол UDP для передачі даних, але додає механізми корекції помилок, такі як ARQ (Automatic Repeat Request) та FEC (Forward Error Correction), щоб забезпечити надійність передачі.

Error Correction: Якщо під час передачі виникають втрати пакетів, SRT використовує ARQ для повторного запиту втрачених пакетів або FEC для відновлення втрачених пакетів на стороні приймача.

Encryption: SRT підтримує шифрування AES для забезпечення безпеки передачі даних.

Режими роботи SRT

- Caller: Відправник ініціює з'єднання з приймачем.
- Listener: Приймач очікує на з'єднання від відправника.
- Rendezvous: Обидва піри одночасно намагаються встановити з'єднання.

Переваги SRT

- Надійна передача даних через ненадійні мережі.
- Підтримка шифрування.
- Простота інтеграції.
- Відкритий протокол.

Недоліки SRT

- Більша затримка порівняно з WebRTC.
- Необхідність встановлення спеціального програмного забезпечення.

Порівняння WebRTC та SRT

Порівняння WebRTC та SRT

Характеристика	WebRTC	SRT
Затримка	Низька	Вища
Надійність	Залежить від мережі	Висока
Шифрування	Вбудоване	Підтримується
Простота	Складна реалізація сигналінгу	Проста інтеграція
Використання	Відеоконференції, онлайн-ігри	Передача відео в реальному часі
Транспортний протокол	UDP (з використанням DTLS та SRTP)	UDP (з ARQ та FEC)
Підтримка браузерів	Так	Ні (потребує спеціального ПЗ)

Рис.3.2 Порівняння WebRTC та SRT

WebRTC та SRT - це дві різні технології, які використовуються для передачі мультимедійних даних в реальному часі. WebRTC підходить для сценаріїв, де важлива низька затримка, таких як відеоконференції та онлайн-ігри. SRT підходить для сценаріїв, де важлива надійність передачі даних,

таких як передача відео в реальному часі через Інтернет. Вибір між WebRTC та SRT залежить від конкретних вимог до проекту.

3.2.9. Переваги обраної архітектури

У цьому документі представлено огляд переваг обраної архітектури для системи, що базується на WebRTC та SRT. Розглянуто ключові характеристики, такі як затримка, стабільність, масштабованість, безпека, гнучкість та розширюваність, а також наведено конкретні результати, яких можна досягти з використанням даної архітектури.

Характеристика та Результати

Обрана архітектура пропонує ряд значних переваг, які роблять її привабливою для реалізації систем реального часу. Нижче наведено детальний опис кожної характеристики та очікуваних результатів.

Затримка

Результат: 70–200 мс (реально досяжно)

Затримка є критично важливим параметром для систем реального часу, особливо для інтерактивних застосунків, таких як відеоконференції та онлайн-ігри. Обрана архітектура дозволяє досягти затримки в діапазоні 70–200 мс. Це досягається завдяки використанню оптимізованих протоколів та ефективній обробці медіа-потоків.

WebRTC: WebRTC розроблено для забезпечення низької затримки, використовуючи UDP для передачі даних та механізми компенсації втрат пакетів.

SRT: SRT (Secure Reliable Transport) також забезпечує низьку затримку, особливо в умовах нестабільної мережі, завдяки механізмам відновлення втрачених пакетів та адаптивному управлінню затримкою.

Стабільність

Результат: адаптивний бітрейт, компенсація втрат

Стабільність є ключовим фактором для забезпечення якісного користувацького досвіду. Обрана архітектура забезпечує стабільність завдяки наступним механізмам:

Адаптивний бітрейт: Система автоматично регулює бітрейт медіа-потоку в залежності від пропускної здатності мережі. Це дозволяє уникнути перевантаження мережі та забезпечити плавне відтворення відео та аудіо.

Компенсація втрат: Використовуються механізми компенсації втрат пакетів, такі як FEC (Forward Error Correction) та ARQ (Automatic Repeat Request), для відновлення втрачених даних та забезпечення безперервного потоку.

Масштабованість

Результат: SFU підтримує сотні клієнтів

Масштабованість є важливою характеристикою для систем, які повинні підтримувати велику кількість користувачів. Обрана архітектура використовує SFU (Selective Forwarding Unit) для ефективного розподілу медіа-потоків між клієнтами.

SFU: SFU отримує медіа-потоки від одного або кількох джерел і пересилає їх лише тим клієнтам, які їх потребують. Це дозволяє значно зменшити навантаження на сервер та забезпечити підтримку сотень клієнтів одночасно.

Горизонтальне масштабування: Можливість додавання нових SFU для збільшення загальної пропускної здатності системи.

Безпека

Результат: DTLS-SRTP, TLS, автентифікація

Безпека є пріоритетом для будь-якої системи, яка передає конфіденційні дані. Обрана архітектура забезпечує безпеку завдяки наступним механізмам:

DTLS-SRTP: WebRTC використовує DTLS (Datagram Transport Layer Security) для шифрування транспортного рівня та SRTP (Secure Real-time Transport Protocol) для шифрування медіа-потоків.

TLS: TLS (Transport Layer Security) використовується для захисту з'єднань між клієнтами та сервером.

Автентифікація: Система підтримує різні механізми автентифікації користувачів для запобігання несанкціонованому доступу.

Гнучкість

Результат: WebRTC → браузери, SRT → професійні клієнти

Гнучкість дозволяє системі підтримувати різні типи клієнтів та інтегруватися з різними платформами. Обрана архітектура підтримує:

WebRTC: WebRTC підтримується більшістю сучасних браузерів, що дозволяє користувачам підключатися до системи без необхідності встановлення додаткового програмного забезпечення.

SRT: SRT використовується професійними клієнтами для забезпечення надійної та якісної передачі відео в умовах нестабільної мережі.

Розширюваність

Результат: можна додати аналітику, хмару, запис

Розширюваність дозволяє додавати нові функції та інтегруватися з іншими системами. Обрана архітектура дозволяє:

Аналітика: Додавання модулів для збору та аналізу даних про продуктивність системи, якість медіа-потоків та поведінку користувачів.

Хмара: Розгортання системи в хмарному середовищі для забезпечення масштабованості та надійності.

Запис: Додавання функціональності для запису медіа-потоків для подальшого відтворення або аналізу.

Обрана архітектура пропонує збалансований набір характеристик, які роблять її привабливою для реалізації систем реального часу. Вона забезпечує низьку затримку, стабільність, масштабованість, безпеку, гнучкість та розширюваність, що дозволяє створювати якісні та надійні застосунки для широкого кола користувачів.

3.3. Реалізація прототипу

Метою реалізації прототипу є створення робочої низьколатентної системи відеонагляду, здатної забезпечити затримку на рівні **менше 200 мс** у реальних мережевих умовах. Архітектура та обрані технології орієнтовані на стабільну роботу в середовищах із нестабільним RTT, пакетними втратами та NAT/CGNAT.

Обґрунтування вибору технологій

Вибір транспортного протоколу

Основним протоколом для передавання відео обрано **WebRTC**, оскільки він:

- підтримує **DTLS-SRTP**, що забезпечує безпечний канал без додаткової конфігурації;
- має **адаптивний бітрейт (Congestion Control)**;
- працює через **NAT/Firewall** завдяки STUN/TURN;
- забезпечує низьку затримку завдяки:
    - мінімальному буферу jitter,
    - NACK/FEC,
    - P2P там, де це можливо.

Для професійного відеопотоку (альтернативний варіант) використовувався **SRT**:

- стабільна передача при високих втратах,
- ARQ-відновлення пакетів,
- ручна регуляція буфера затримки.

Вибір кодека

Використано **H.264 (AVC)** у режимі:

- **ultrafast preset**,
- **low-latency tune**,
- **IBBP або IPPP GOP = 15–30 кадрів**,
- **CAVLC замість CABAC**, якщо доступно.

Переваги:

- мінімальна затримка кодування/декодування – 5–15 мс;
- підтримка більшістю камер і браузерів;
- апаратне декодування на GPU (VAAPI/NVDEC, VideoToolbox).

Чому не HEVC (H.265)?

- HEVC дає кращу якість, але має **декодування довше** на 40–80%,
- браузери обмежено підтримують H.265,
- для low-latency критичний саме час.

3.3.2. Реалізація серверної частини

Сервер складається з двох підсистем:

1. **Сервер сигналізації**, що керує WebRTC-сеансами.
2. **SFU-сервер**, що маршрутизує та передає потоки.

Сигналізація

Для сигналізації використано:

- WebSocket (WS/WSS),
- обмін SDP-описами,
- обробку ICE-кандидатів,
- авторизацію доступу до камер.

Основні функції сервера сигналізації:

- створення peer-сесії,
- авторизація, токени доступу,
- зберігання параметрів камери,
- передача SDP між клієнтом та edge-модулем.

Медіасервер (SFU)

У прототипі використано:

- **Mediasoup** або **Janus WebRTC Server**.

Функції SFU:

- приймає потік від Edge-модуля,
- мінімізує буфер (20–40 мс),
- перенаправляє потоки до клієнтів,
- забезпечує NACK/FEC,
- динамічно коригує бітрейт.

Реалізовано:

```
SFU → jitter buffer = 20 ms
     → packet retransmission (NACK)
     → codec negotiation (H.264 baseline)
     → RTP priority = high
```

У режимі SRT сервер виконує роль **SRT Listener** або **Relay**.

3.3.3. Реалізація клієнтської частини

Клієнт реалізовано у двох варіантах:

- браузерний інтерфейс (JavaScript WebRTC API),

- мобільний застосунок (Android із WebRTC Native API).

Основні функції клієнта:

1. **Прийом відеопотоку (RTCPeerConnection)**
2. **Декодування апаратними засобами**
    - Chrome → GPU VideoDecoder API
    - Android → MediaCodec
3. **Мінімальна клієнтська буферизація**
    - 1–2 кадри у відеоплеєрі,
    - вимкнено згладжування кадрів.
4. **Отримання статистики WebRTC**
    - `inbound-rtp` FPS,
    - RTT,
    - бітрейт,
    - packet-loss.

Інтерфейс

Інтерфейс включає:

- перегляд відео в реальному часі,
- показ затримки,
- вибір режимів якості,
- перемикання камер.

3.3.4. Кодування відео (H.264 Low-Latency Mode)

Параметри кодування Edge-модуля

Використано FFmpeg або GStreamer із параметрами:

```
ffmpeg -hwaccel vaapi -i rtsp://camera \
```

```
-c:v h264_vaapi \
-preset ultrafast \
-tune zerolatency \
-g 30 \
-bf 0 \
-b:v 2M \
-maxrate 2M \
-bufsize 0 \
output_webrtc_stream
```

Пояснення:

- **-tune zerolatency** — мінімізація буферів на рівні енкодера,
- **-g 30** — короткий GOP (1 I-frame кожні ~1 сек),
- **-bf 0** — без B-frames, щоб уникнути затримки декодування,
- **bufsize 0** — вимкнення VBV.

Результат:

- затримка кодування → **5–12 мс**,
- рівномірний бітрейт,
- плавна робота при нестабільних каналах.

3.3.5. Оптимізація буферів

1) Jitter-buffer (Edge-модуль)

- базовий обсяг → **20–40 мс**,
- адаптивне розширення до 60–80 мс при втраті пакетів.

2) Буфер SFU

- жорстке обмеження → **<= 20 мс**,

- відкидання застарілих кадрів,
- обов’язкова синхронізація RTP timestamp.

3) Буфер клієнта

- обмежено до **1–2 кадрів**,
- JavaScript-програвач працює у відключеному режимі sync-playout.

Буфер RTCP

- інтервал report → 100–200 мс,
- прискорене реагування на втрати.

4) TURN буфер

Якщо використовується TURN:

- додається 10–40 мс,
- у прототипі використано relay-режим з мінімальним internal buffer.

3.3.6. Підсумкові показники прототипу

Параметр	**Значення**
Затримка "камера → клієнт"	**90–180 мс**
Роздільність	720p або 1080p
Кодек	H.264 baseline/ main
FPS	25–30
Надійність при packet-loss	до 8–10% завдяки NACK/FEC / SRT ARQ
Підтримка NAT	STUN/TURN, WebRTC ICE

Висновки до розділу 3

У третьому розділі було здійснено повний цикл проєктування та розробки прототипу системи відеонагляду з низькою затримкою, орієнтованої на роботу в мережах із нестабільною пропускною здатністю та підвищеними вимогами до оперативності передачі відеосигналу. На основі сформульованих вимог було розроблено архітектуру, яка забезпечує затримку на рівні **менше 200 мс**, високу стабільність та надійність передавання потоків у реальному часі.

Було визначено ключові технології та методи, що дозволяють досягти низьколатентної роботи системи. Зокрема, використання протоколу **WebRTC** забезпечило адаптивний контроль пропускної здатності, підтримку NAT traversal через **STUN/TURN**, а також шифрування трафіку за допомогою DTLS-SRTP. Як альтернатива для професійних застосувань запропоновано протокол **SRT**, який демонструє високу стійкість до втрат пакетів. Практичні експерименти показали, що такі протоколи є оптимальними для побудови систем відеонагляду з мінімальною затримкою.

У межах реалізації прототипу було створено серверну інфраструктуру, яка включає модуль сигналізації та медіасервер типу SFU, що виконує маршрутизацію потоків та мінімізацію буферизації. Розроблено клієнтську частину, яка використовує апаратне декодування відео та оптимізовані режими програвання для зменшення затримки. Особливу увагу приділено кодуванню відео за допомогою **H.264 у режимі low-latency**, що дозволило скоротити час кодування та стабілізувати передавання.

Важливим елементом дослідження стала оптимізація буферів на всіх етапах проходження відеопотоку — від edge-модуля та SFU до клієнтського програвача. Мінімізація розміру jitter buffer, скорочення фінальних буферів відтворення та жорсткий контроль RTP-синхронізації забезпечили

зменшення загальної енд-то-енд затримки до рівня **90–180 мс**, що повністю задовольняє вимоги до системи.

Таким чином, розроблена система довела можливість побудови ефективного низьколатентного відеонагляду на основі сучасних протоколів реального часу та оптимізованих схем кодування. Архітектура є масштабованою, безпечною та придатною для розгортання у реальних умовах, що створює основу для подальшого розвитку та впровадження розширених функцій — аналітики відео, автоматичного виявлення подій та інтелектуальної обробки потоку.

ВИСНОВОК

У магістерській роботі було вирішено актуальне науково-практичне завдання, пов’язане з розробкою системи відеонагляду з низькою затримкою, здатної забезпечувати оперативне отримання відеоінформації в умовах обмеженої пропускної здатності мережі, наявності джитера та пакетних втрат. Актуальність теми зумовлена високими вимогами сучасних систем безпеки до швидкодії, стабільності та достовірності відеосигналу, що особливо критично для сценаріїв моніторингу, управління в режимі реального часу та реагування на події.

У першому розділі виконано аналітичний огляд сучасних технологій відеонагляду, охарактеризовано архітектуру систем, методи передавання відеопотоків і фактори, що впливають на затримку. Проведено порівняння популярних протоколів — RTSP, RTMP, WebRTC, SRT, HLS — з точки зору їх придатності до використання у low-latency рішеннях. Проаналізовано компроміси між затримкою, якістю зображення та надійністю передавання. Також розглянуто комерційні рішення, які демонструють мінімальну енд-то-енд затримку на ринку 2024–2025 років.

У другому розділі сформовано теоретичний фундамент системи низької затримки. Розроблено математичні моделі відеопотоку, включно з моделлю буферизації, моделлю затримки кадрів та розрахунком необхідної пропускної здатності каналу. Проаналізовано транспортні технології, що забезпечують мінімальну затримку, зокрема UDP, QUIC, SRTP/DTLS, а також особливості їхнього застосування у відеопотоках. Розглянуто алгоритми адаптивного керування мережею, методи компенсації втрат і оптимізації структури GOP під потреби реального часу.

У третьому розділі представлено проєктування та реалізацію прототипу системи відеонагляду з низькою затримкою. Розроблено архітектуру, що включає камеру, edge-модуль, медіасервер та клієнтську частину, які взаємодіють через протокол WebRTC або SRT. Реалізовано серверну інфраструктуру, технології сигналізації, маршрутизації потоків, а також оптимізоване кодування відео у режимі low-latency. Особливу увагу приділено мінімізації буферизації на всіх етапах проходження відеопотоку та забезпеченню стабільної роботи системи в умовах нестабільної мережі. Експериментальні результати підтвердили досягнення цільових показників — енд-то-енд затримки на рівні **90–180 мс**, що відповідає вимогам до сучасних низьколатентних систем відеоспостереження.

Підсумовуючи результати виконаної роботи, можна стверджувати, що мета дослідження — розробка ефективної системи відеонагляду з низькою затримкою — успішно досягнута. Розроблена система є масштабованою, безпечною та адаптивною до різних мережевих умов. Вона може бути основою для практичного впровадження у сферах безпеки, робототехніки, віддаленого керування, інтелектуального моніторингу тощо. Отримані результати створюють перспективу для подальших досліджень, зокрема у напрямках інтеграції відеоаналітики, машинного навчання, розвитку edge-обчислень та підвищення автономності систем реального часу.