МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о. завідувача кафедри
_____ Сафонова С.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження технологій для кросплатформного обміну мультимедійним контентом

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 122 "Комп'ютерні науки "

Науковий керівник роботи: _____ (підпис) М.Є. Щербакова (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О. Критська (ініціали, прізвище)

Студент: _____ (підпис) С.А. Покришка (ініціали, прізвище)

Група: КН -18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень "магістр"
Спеціальність 122 – "Комп'ютерні науки"
(шифр і назва)
Спеціалізація
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о.зав. кафедри *КНІ*
д.т.н., доц. *С.О. Сафонова*
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Покришці Сергію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження технологій для кросплатформного обміну мультимедійним контентом

керівник проекту (роботи) Щербакова М.Є., к.т.н., доцент
(прізвище, ім.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «11» 10 2019 р. № ____
2. Строк подання студентом роботи 15.01.2020
3. Вихідні дані до роботи Матеріали науково-дослідної практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз існуючих технологій для кросплатформного обміну мультимедійним контентом
2. Дослідження та порівнянння технологій між собою
3. Розробка додатку для обміну мультимедійним контентом
4. Запропонувати метод для покращення показників затримки та якості медіа
5. Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Критська Я.О.*		

7. Дата видачі завдання *06.09.2019*

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання, збір матеріалів	*10.09.19-17.09.19*	
2	Огляд літератури й обґрунтування необхідності дослідження	*18.09.19-25.09.19*	
3	Дослідження методів та технологій для передачі мультимедійного контенту	*26.09.19-18.10.19*	
4	Аналіз та порівняння технологій	*19.10.19-06.11.19*	
5	Розробка веб - додатку	*17.11.19-08.12.19*	
6	Охорона праці	*09.12.19-15.12.19*	
7	Оформлення пояснювальної записки	*16.12.19-29.12.19*	
8	Підготовка та подання магістерської роботи до захисту	*03.01.20-12.01.20*	

Студент _____ *Покришка С.А.*
(підпис) (прізвище та ініціали)

Науковий керівник _____ *Щербакова М.Є.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Покришка С.А. Дослідження технологій для кросплатформного обміну мультимедійним контентом

В роботі розглянуто принципи для передачі інформації в мережах, основні технології для кросплатформної передачі мультимедійного контенту, проаналізовано ці технології та проведено дослідження і порівняння, після чого була обрана краща технологія для передачі контенту та проведено більш детальний аналіз як можливостей, так і алгоритмів для отримання, обробки та передачі даних.

Розроблено веб – додаток для кросплатформного обміну мультимедійним контентом. Проведені тести.

Ключові слова: RTMP, RTP, SDP, java, flash, webRTC, мовлення.

АННОТАЦИЯ

Покрышка С.А. Исследование технологий для кроссплатформенного обмена мультимедийным контентом

В работе рассмотрены принципы для передачи информации в сетях, основные технологии для кроссплатформенной передачи мультимедийного контента, проанализированы эти технологии и проведено исследование и сравнение, после чего была выбрана лучшая технология для передачи контента и проведён более детальный анализ как возможностей, так и алгоритмов для получения, обработки и передачи данных.

Разработан*о* веб - приложение для кроссплатформенного обмена мультимедийным контентом. Проведены тесты.

Ключевые слова: RTMP, RTP, SDP, java, flash, webRTC, вещание.

ABSTRACT

Pokrishka S. Research of technologies for cross-platform exchange of multimedia content

The paper discusses the principles for information transmission on networks, the basic technologies for cross-platform transmission of multimedia content, analyzes these technologies and conducts research and comparison, then selects the best technology for content transmission and provides a more detailed analysis of both the capabilities and algorithms for obtaining, data processing and transmission. A web application for cross-platform multimedia content sharing has been developed. Conducted tests.

Keywords: RTMP, RTP, SDP, java, flash, webRTC, broadcast.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML5 - HyperText Markup Language, version 5

UDP - User Datagram Protocol

RTSP – Real Time Streaming Protocol

RTP – Real-time Transport Protocol

RTCP - Real-Time Transport Control Protocol

TCP - Transmission Control Protocol

IGMP - Internet Group Management Protocol

PIM - Protocol Independent Multicast

VoD - Video on demand

WebRTC - Real-time communications

IETF - Internet Engineering Task Force

ВСТУП

Нині спостерігається активна інформатизація та цифровізація у всіх сферах життя. Відповідно до такої тенденції стрімкими темпами зростає і число інформаційних проектів, програмного забезпечення, інструментів розробки програмного забезпечення, мов програмування і інфраструктурних технологій, зростає і кількість інформації, якою обмінюються користувачі. Геоінформаційні технології також не стоять на місці. З'являється безліч просторових даних, які потрібно якимось чином зберігати, обробляти і налагоджувати обмін цією інформацією між користувачами і виробниками. Слідом за цим зростає число геоінформаційних систем, веб-картографічних сервісів та інших сервісів, які вирішують завдання, пов'язані з просторовими даними.

Ключова роль в обміні просторовими даними і взагалі даними в наші дні належить мережі Інтернет. Сучасна цифрова інфраструктура просторових даних, в більшості випадків, має на увазі централізоване зберігання даних на серверах в базах даних або інших сховищах інформації. Такий метод зберігання і обробки має свої переваги і недоліки. До переваг даного методу можна віднести єдину точку доступу до ресурсів, централізований контроль доступу до даних, а також гнучкість в створенні резервних копій баз даних та реалізації інших методів збереження даних.

Останнім часом кількість просторової інформації зросла по експоненціальному закону у зв'язку з розвитком систем дистанційного зондування Землі, що включає в себе супутникові знімальні комплекси, атмосферні методи дослідження (аерофотозйомка), а також розвитком глобальної мережі Інтернет і сервісів обміну та обробки просторових даних. Це зростання показало кілька недоліків централізованого зберігання даних:

1. збільшення витрат на підтримку серверів для обробки і зберігання даних;
2. зниження номінальної швидкості обміну інформацією між сервером і клієнтами за рахунок безпосередньо зростання кількості інформації, що підлягає обміну;
3. уразливості сервера і сховища даних перед хакерськими атаками, збільшення навантаження на канал передачі даних при зростанні числа користувачів сервісу;
4. використання спеціалізованої фізичної серверної інфраструктури (датацентри).

Актуальність теми. Комунікації в реальному часі дуже чутливі до мережі: пропускної здатністності, затримки і втрати пакетів. Зниження бітрейту веде до зниження якості відео, тривала мережева затримка веде до тривалої затримки у кінцевих користувачів. Втрата пакетів може зробити звук переривчастим і привести до фризів на відео (через пропуск кадрів). Тому дуже важливо обрати оптимальну технологію для спілкування та передачі файлів.

Мета і задачі дослідження: аналіз, дослідження та порівняння основних технологій для вибору оптимального варіанту для розробки.

Об'єкт дослідження: протоколи та технології для кросплатформного обміну мультимедійним контентом.

Предмет дослідження: технології для передачі мультимедійного контенту java, flash та webrtc.

Методи дослідження: порівняння доступності та можливостей технологій. Також були застосовані теоретичні (аналіз предметної області, збір необхідних даних для реалізації поставленої мети), емпіричні (експертні оцінки тестування та вибір засобів розробки) та статистичні методи дослідження.

Наукова новизна одержаних результатів полягає в наступному:

Запропоновано метод для покращення масштабованості і якості медіа в WebRTC-додатках, використовуючи каскадування SFU (вибірковий направляючий блок).

Розроблено архітектуру програмного забезпечення, особливістю якої є впровадження каскадування SFU.

Практичне значення одержаних результатів полягає у наступному: запропонований метод дозволяє зменшити середню затримку між абонентами при передачі медіа контенту.

Апробація результатів роботи. Основні положення магістерської роботи доповідалися та обговорювалися на Всеукраїнській науково-практичній конференції з міжнародною участю «Майбутній науковець-2019» (м. Сєвєродонецьк, 12 грудня 2019 р.) та на V Молодіжному форумі «ІТ-Ідея» (м. Сєвєродонецьк, 6 грудня 2019 року).

Публікації. За темою магістерської роботи з викладенням її основних результатів опубліковано 1 статтю в науковому фаховому виданні України, 2 тез доповідей.

Структура та обсяг магістерської роботи. Магістерська робота складається зі вступу, 4 розділів, висновків, переліку посилань до розділів з 34 найменувань, додатку на 9 сторінках. Загальний обсяг роботи складає 91 сторінку. Магістерська робота містить 40 рисунків та 6 таблиць.

РОЗДІЛ 1
ОГЛЯД ПРОТОКОЛІВ ТА ТЕХНОЛОГІЙ ДЛЯ ПЕРЕДАЧІ ІНФОРМАЦІЇ

Розглянуто основні протоколи та технології для передачі інформації, описано основні проблеми.

1.1 Загальні відомості про потокове мультимедіа

Потокове мовлення - це мультимедійний контент, який користувачі безперервно отримують від провайдера послуг. Термін в тому числі застосуємо як до таких джерел передачі інформації, як радіо і телебачення.

Передача потокового відео використовує технології стиснення і буферизації даних, завдяки чому трансляція ведеться в режимі реального часу при відправці потоку як послідовності стислих пакетів. Особливість такого методу в тому, що користувачеві не потрібно чекати повного завантаження відео файлу для того, щоб почати перегляд.

Для перегляду потокового відео на пристрої користувача повинен бути встановлений спеціальний відеоплеєр, що підтримує цю функцію. До речі, що поставляється в комплекті стандартного набору ПО Windows Media Player відноситься до таких. Серед інших популярних програм варто відзначити Quicktime Player і RealOne Player.

Однак відсутність програми на ПК не обмежує користувача: в інтернеті є безліч ресурсів, які підтримують потокове онлайн мовлення. Найвідомішими з них є Twitch і Youtube. Останнім часом технологію підхопили популярні соціальні мережі - наприклад, Facebook і Вконтакте.

Однак відсутність програми на ПК не обмежує користувача: в інтернеті є безліч ресурсів, які підтримують потокове онлайн мовлення. Найвідомішими з них є Twitch і Youtube. Останнім часом Facebook і Вконтакте.

Для трансляції потокового мультимедіа зазвичай використовується один з двох способів:

Послідовний - відеофайл відтворюється з жорсткого диска комп'ютера користувача або сервера провайдера послуг. Як правило, при передачі таким способом якість зображення і звуку вище. До недоліків варто віднести те, що неможливо переключити ролик з одного моменту на інший, не дочекавшись його буферизації. Тобто, потрібний фрагмент

повинен бути завантажений, щоб відтворити його для користувача переглянути. Для трансляції підходить звичайний веб-сервер.

У реальному часі - вимагає наявності потокового сервера. Цей метод більше підходить для передачі відеофайлів великої тривалості. Користувач може вибрати місце, з якого він хоче почати перегляд. Також цей вид потокового мультимедіа мовлення використовується для трансляції з веб-камери або захоплення екрану.

1.2 Потокове мовлення і зберігання інформації

Розмір, який потрібен для зберігання потокового мультимедіа контенту (в більшості файлових систем виражається в мегабайтах, гігабайтах, терабайт і т. д.) обчислюється в залежності від швидкості інформації, яку передають, та тривалості інформації за наступною формулою:

обсяг сховища (в МБ) = загальна тривалість (в сек.) * бітрейт (в кбіт / с) / (8 * 1024)

Одна година відео, що транслюється зі швидкістю 6000 кбіт / с (типове відео, яке транслюється в інтернеті, що має розмір 1920x1080 пікселів) буде займати:

(3,600 с * 6000 кбіт / с) / (8 * 1024) буде займати 2636 мегабайт (близько 2,5 гігабайт)

Якщо файл, який зберігається на сервері з режимом передачі за запитом буде дивитися близько 1000 людей одночасно, використовуючи протокол Unicast (1 клієнт - 1 з'єднання), то сервер повинен мати наступну пропускну здатність:

6000 кбіт / с * 1,000 = 6,000,000 кбіт / с = 6 Гбіт / с мережевого інтерфейсу

Це еквівалент близько 3,5 ТБ інформації на годину.

1.3 Протоколи потокового мовлення

Розробка протоколів потокового мовлення має наступні проблеми:

- Датаграмні протоколи, наприклад User Datagram Protocol, відправляють порцію медіаконтенту як потік окремих маленьких пакунків. Він легкий і ефективний; в той же час,

в специфікації протоколу немає системи для гарантованої отримки даних. Це дуже сильно ускладнює пошук і виправлення одержуваних даних. При втраті даних потік може бути відключений.

- Протоколи RTSP, RTP і RTCP спеціально були розроблені для передачі мультимедійної інформації по мережі. Останні два основані на UDP.

- Надійні протоколи, наприклад TCP, гарантують правильність одержуваних даних клієнтів мовлення. Але при великій кількості помилок при з'єднанні чи підтвердженні одержуваної інформації, інформація може стати неактуальною. Це може викликати великі затримки при передачі інформації на час, який було затрачено на пересилку пошкодженої інформації. Одним із варіантів рішення проблеми є буферизація інформації на стороні клієнта.

- Протоколи Unicast відправляють копію даних кожному клієнту. Unicast підходить для більшості рішень в мережі Інтернет, але дуже ускладнює масштабування сервера для великої кількості клієнтів.

- При широкомовній передачі одна копія даних передається всім клієнтам сервера.

Протоколи Multicast розроблені для того, щоб знизити навантаження з серверів на підключення / ширину каналу при передачі потокового мультимедіа великій кількості клієнтів. Ці протоколи відсилають одну порцію даних одразу цілій группі клієнтів. Залежно від типу мережі, групова передача даних може бути доступна, а може і не доступна. Одним з потенційних недоліків групової передачі є відсутність можливості розробити функцію відео за запитом. Потокове мовлення інформації також унеможливлює управління відтворенням. Однак, ця проблема може бути вирішена впровадженням в мережу передачі даних кешуючих серверів і буферизуючого потік програмного забезпечення.

- Multicast дає можливість передавати один потік інформації группі клієнтів по мережі. Однією із проблем при розробці подібного рішення потокового мовлення є правильна настройка маршрутизаторів для передачі пакетів з однієї частини мережі в іншу. Якщо заклад, що надає потокове мовлення, має контроль над мережею між сервером і клієнтами, то протоколи, такі як IGMP і PIM, можуть бути використані для відправки мультимедіа контенту кільком клієнтам із різних мереж LAN.

- Протоколи P2P можуть бути використані при поширенні попередньо збереженої мультимедіа між комп'ютерами. Це дозволяє уникнути навантаження сервера, однак мережа передачі даних між сервером і одним з клієнтів стає слабким місцем даного способу реалізації потокового мовлення інформації.

1.4 Системи відеопередачі в тимчасових розподілених мережах

На сьогоднішній день сервіси VoD [1], такі як YouTube, генерують велику кількість трафіку в мережі Інтернет, до того ж популярність таких сервісів тільки зростає. Тому, для зменшення навантажень, провайдери сервісів зберігання медіа-файлів все частіше починають використовувати розподілені мережі. Мовлення в тимчасових мережах є перспективною альтернативою, яка може задовольнити зростаючі вимоги. Стандарти HTML5 [2] і WebRTC надають інструменти, що дозволяють браузерам взаємодіяти один з одним безпосередньо в режимі реального часу. Проте, створення системи потокового мовлення в тимчасових мережах з використанням HTML5 може бути проблематичним. Так, стандарти HTML5 та WebRTC ще остаточно не затверджені, і тому браузери мають дуже слабку підтримку цих технологій, вони працюють нестабільно і ненадійно.

Система відеопередачі в тимчасових розподілених мережах є хорошим прикладом складної системи, яка вимагає цілий комплекс локальних операцій із завантаження, зберігання, відтворення і обміну даними між вузлами мережі.

Існує кілька варіантів реалізації такої системи. Одним з них є варіант заснований на технологіях HTML5, WebRTC і BitTorrent.

Стандарт HTML5 включає в себе багато нововведень і дозволяє будувати складні веб-додатки з можливостями паралельної обробки даних, комунікацій в реальному часі (realtime communication - RTC) і зберігання великих обсягів даних на клієнті. Всі ці можливості доступні через публічні прикладні програмні інтерфейси (API), які браузер викликає за допомогою мови JavaScript .

WebRTC - це стандарт, який дозволяє веб-додаткам встановлювати пряме з'єднання між двома браузерами без необхідності передавати дані через веб-сервер. Даний стандарт складається з публічного API і набору протоколів для передачі даних між браузерами. Стандарт WebRTC дозволяє комунікацію між браузерами в реальному часі, використовуючи протокол UDP в парі з протоколом TCP для звичайних HTTP-запитів. Цей стандарт визначає три режими і API для комунікації між браузерами: Peer Connection API для встановлення з'єднання між вузлами мережі, Media Stream API для потокового відеовещанія і Data Channel API для передачі інших даних.

BitTorrent [3] - це протокол для обміну файлами в тимчасових мережах, який використовується для поширення даних в мережі Інтернет. Для того щоб відправити або отримати файл використовується BitTorrent клієнт - комп'ютерна програма, яка підтримує роботу з протоколом BitTorrent. Зазвичай при передачі даних за допомогою протоколу

BitTorrent, файли розбиваються на частинки, які передаються між вузлами тимчасової мережі у випадковому порядку. Але для реалізації потокового мовлення порядок цих частинок є важливим, саме тому така система повинна також враховувати цей фактор. Іншою проблемою є низька підтримка протоколу BitTorrent сучасними браузерами, тому для більшості браузерів необхідно встановлювати спеціальні розширення для підтримки BitTorrent.

1.5 Еволюція браузерних технологій

Спочатку інтернет працював на HTTP запитах [4]. Звичаний web запит виконаний на сторінці, що повертає код сторінки. В 2004 році була розроблена та впроваджена технологія AJAX, яка дозволяла оновлювати частину сторінки без перезавантаження її цілком. Потім в 2008 році була розроблена технологія Web Socket. Це дало можливість відкривати сесію для двонаправленого спілкування між клієнтом та сервером. І вже в 2012 році була розроблена технологія WebRTC, за допомогою якої сторінка браузера може обмінюватись інформацією напряму з другою сторінкою.

1.6 Однорангові і багаторангові мережі

Існують дві основні моделі мережевих взаємодій комп'ютерної техніки. Залежно від того як розподілені функції між вузлами учасниками мережі виділяють однорангові і багаторангові мережі. У тимчасових або пірінгових (від англ. Peer-to-peer) мережах все агенти спілкування рівноправні і тому немає переваг у будь-кого з учасників. Такі мережі і є основою всіх децентралізованих архітектур мережевого обміну інформацією. Комп'ютери-учасники цих мереж можуть одночасно виступати клієнтами і виконувати запити до інших вузлів мережі і можуть виконувати функції сервера, отримуючи запити від інших учасників мережі, обробляючи ці запити і генеруючи відповіді. Структура такої мережі показана на рис.1.1.

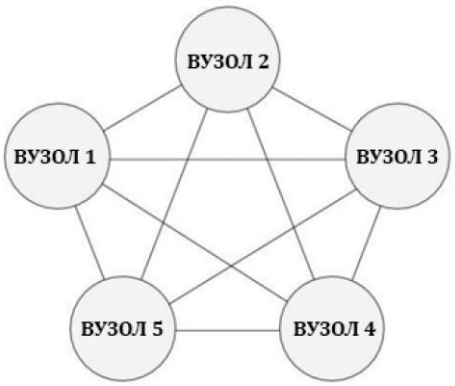


Рисунок 1.1 - Структура однорангової мережі

На відміну від тимчасових мереж, багаторангові в своїй структурі одного або декілька вузлів мережі з правами, більшими, ніж у інших учасників комунікації. Найпоширенішою архітектурою є клієнт-серверна архітектура. Вона включає в себе два рангу учасників спілкування, ранг клієнт і ранг сервера, при цьому ранг сервера є чільним в мережі. Ця модель - основа для централізованого обміну і зберігання інформацією, і найпоширеніша мережева архітектура в сучасній мережі Інтернет. Структура цієї мережі показана на рисунку 1.2. З рисунка видно, що "Вузол 1" є сервером і до нього звертаються із запитами всі клієнти мережі. Також видно, що інші вузли один з одним не спілкуються безпосередньо, у них не доступна така можливість. Всі міжклієнтні взаємодії відбуваються або через посередництво сервера, або не відбуваються взагалі. При цьому при виході з ладу сервера виходить з ладу вся мережа і вузли-клієнти абсолютно втрачають можливість в отриманні послуги, що надається вузлом-сервером.

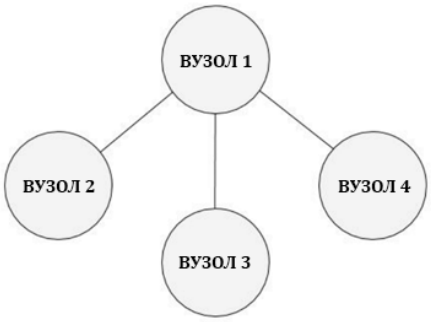


Рисунок 1.2 - Структура багаторангової мережі

Резюмуючи вищесказане, очевидними перевагами володіє однорангова мережа в умовах великої кількості даних і користувачів, знижуючи навантаження на вузол.

1.7 Існуючі розподілені методи зберігання, обробки та обміну інформацією

Незважаючи на чільне місце клієнтсерверной (багатораннгової) архітектури в сучасній мережі Інтернет існують децентралізовані моделі комунікацій, в яких немає єдиного сервера або іншого авторитарного агента, провідного нагляд за мережею. До таких мереж відносяться Torrent-трекери, blockchain-мережі, тіньовий інтернет. Розглянемо докладніше один з методів.

Протокол BitTorrent розроблений спеціально для обміну великими файлами через мережу Інтернет. Являє собою реалізацію архітектури тимчасової мережі. Типова мережа BitTorrent складається з безлічі клієнтів, які будуть здійснювати обмін файлами і так званого торрент-трекера - сервера, який здійснює відстеження за поточними торрент-роздачами файлів і надає користувачам мережі інформацію про інших користувачів і файлах, які можна завантажити. Структура обміну файлами відбувається за наступним механізмом:

1. Користувач 1, який хоче опублікувати файл, за допомогою спеціального програмного забезпечення створює torrent-файл, який містить метаінформацію про опублікованому ресурсі, а також дані містять мережеві параметри підключення до Користувача 1

2. Користувач 1 викладає створений файл на спеціальній платформі - торрент-трекер або створює роздачу і зазвичай супроводжує дану роздачу семантичним описом

3. Користувач 2 знаходить на торрент-трекер роздачу Користувача 1, викачує торрент-файл і відкриває його в спеціальному програмному забезпеченні, починається скачування безпосередньо файлу безпосередньо з комп'ютера Користувача 2

4. Користувач 3 знаходить на торрент-трекер роздачу Користувача 1, викачує торрент-файл і відкриває його в спеціальному програмному забезпеченні, починається скачування файлу шматками і з комп'ютера Користувача 1 і з комп'ютера користувача 2 Файл Користувача 1 таким чином копіюється на клієнтах, запитали цей файл і далі стає можливим скачування цього файлу зі всіх сайтів у цій мережі, що запитали і скачали хоча б невеликий фрагмент даного файлу. Переваги очевидні - знижується навантаження на канал між користувачами, швидкість обміну інформацією зростає

1.8 Основні проблеми p2p з’єднання

Встановити з'єднання p2p - досить важке завдання, так як комп'ютери не завжди володіють публічними IP адресами, тобто адресами в інтернеті. Через невеликої кількості IPv4 адрес (і для цілей безпеки) був розроблений механізм NAT, який дозволяє створювати приватні мережі, наприклад, для домашнього використання. Переважна кількість домашніх роутерів зараз підтримують NAT і завдяки цьому всі домашні пристрої мають вихід в інтернет, хоча провайдери інтернету зазвичай надають одну IP адресу. Публічні IP адреси - унікальні в інтернеті, а приватні ні. Тому з'єднатися p2p - важко. Наприклад, коли вузли в різних мережах, або коли у них співпадають внутрішні адреса.

1.9 Технології комунікації реального часу

Робота над забезпеченням можливості комунікацій реального часу (Real-Time Communications, RTC) в WWW здійснюється в двох основних організаціях по стандартизації Мережі: IETF і консорціумі World Wide Web Concortium.

Завдання стандартизації в цій галузі полягає в створенні універсального інтерфейсу програмування, який дозволить веб-додатку, що працює на будь-якому пристрої, з використанням захищеного доступу до веб-камер і мікрофонів в режимі реального часу пересилати будь-які дані між браузерами по одноранговому принципу. Такий API повинен дозволити розробникам реалізовувати функції пошуку контактів, створення сеансу зв'язку з ними і буде покладатися на існуючі специфікації, визначені IETF як найбільш придатні для вирішення мережевих завдань: керуючі протоколи, протоколи встановлення з'єднання і управління ним, транспортні протоколи без встановлення фізичного з'єднання, кодеки і т. д. Між процесами стандартизації, що проходять в IETF і в W3C, немає чіткої межі - цим двом організаціям неминуче доводиться взаємодіяти на перетині облас їй, одна з яких пов'язана з функціями комунікаційного додатки, виконуваного на окремому вузлі, інша - з обміном інформацією між сторонами.

Перехід на міжбраузерний зв'язок реального часу представляється серйозним проривом в індустрії, і сьогодні ця тема активно обговорюється в IT-співтоваристві.

1.10 Ідея міжбраузерного зв'язку

Міжбраузерний обмін в реальному часі пропонується замість традиційних комунікаційних сервісів. Наприклад, в одному дослідженні оцінювалася складність традиційних телекомунікаційних систем та веб-архітектур, і фахівці прийшли до висновку, що дві ці області слід зблизити, щоб комунікації в реальному часі стали доступні якомога більшій кількості користувачів. Ці дослідження лягли в основу однієї специфікації, яка описує REST-інтерфейс для протоколу SIP (Session Initiation Protocol). Дослідники і розробники ініціювали чимало інших аналогічних проектів, проте часто без опори на стандарти (наприклад, за рахунок використання пропрієтарних плагінів як Adobe Flash і Microsoft ActiveX) і без складання документації. Виняток - робота дослідників з Ericsson Labs, в якій була зроблена спроба реалізації вбудованих засобів підтримки протоколу Real-Time Transport Protocol і медіапристроїв в браузерах за рахунок спеціального API JavaScript, зміненого варіанту движка WebKit і фреймворка Gstreamer.

Всі ці ініціативи підштовхнули розробників стандартів Інтернету зайнятися вирішенням даного завдання, що в кінцевому рахунку призвело до створення двох взаємопов'язаних робочих груп: RTCWeb в IETF і WebRTC (Web Real-Time Communication) в W3C. Перша займається протоколами і взаємодіями, що відносяться до ведення IETF, в тому числі інтероперабельністю зі старими системами (наприклад, з існуючими телекомунікаційними), а група WebRTC розробляє API, що дозволяє браузерам і скриптовою мов взаємодіяти з медіапристроїв (мікрофонами, веб-камерами і динаміками), засобами обробки (кодеками) і функціями передачі. Дані ініціативи, швидше за все, приведуть до розширення специфікації HTML5, в якій вже є стандартний спосіб потокової передачі мультімедіаконтента з серверів на браузери.

1.11 Відкрита веб – платформа

Під HTML5 зазвичай розуміється все, що узагальнює досягнення в області так званої відкритої платформи WWW. Однак сам по собі HTML несе лише частину технологій, що використовуються для розробки веб-додатків, які, як традиційно вважається, становлять відкриту платформу для Web. У число цих технологій також входять Cascading Style Sheets (CSS), Document Object Model, JavaScript і ряд скриптових інтерфейсів програмування.

HTML служить для структурованого уявлення призначеного для користувача інтерфейсу і даних програми. Розробники можуть стилізувати додаток за допомогою CSS і

керувати ним за допомогою коду JavaScript. Всі три технології передаються по інфраструктурі Web (через проксі, веб-сервери і браузери) по протоколах HTTP або Web-Socket.

API для скриптів дозволяють програмістам використовувати можливості браузера і керувати ним за допомогою JavaScript. Зазвичай, як тільки в браузерах з'являються нові особливості, в W3C розробляють нові API, що надають доступ до цих функцій. В результаті браузери наближаються до середах виконання нативних додатків.

1.12 Архітектурна модель RTC

Архітектурна модель RTC (рис. 1.3) дозволяє пересилати медіа-дані безпосередньо між браузерами без посередництва серверів. Маршрут сигнальних даних, однак, проходить через сервери, які можуть змінювати, ретранслювати сигнали або керувати ними в міру необхідності. Ідея в тому, щоб клієнтські веб-додатки (зазвичай написані на суміші HTML і JavaScript) могли взаємодіяти з браузерами через API WebRTC як в проактивному режимі (наприклад, для опитування можливостей браузера), так і в реактивному (для отримання згенерованих браузером повідомлень). Такий API надасть широке коло функцій, включаючи: управління з'єднанням (по тимчасової схемою); засоби кодування / декодування; механізми управління сеансом; засоби управління медіаданими; функції брандмауера і механізм обходу систем трансляції мережевих адрес (Network Address Translation, NAT). Даний API реалізується на JavaScript, який довів свою ефективність як потужний скриптова мова для клієнтської частини веб-додатків.

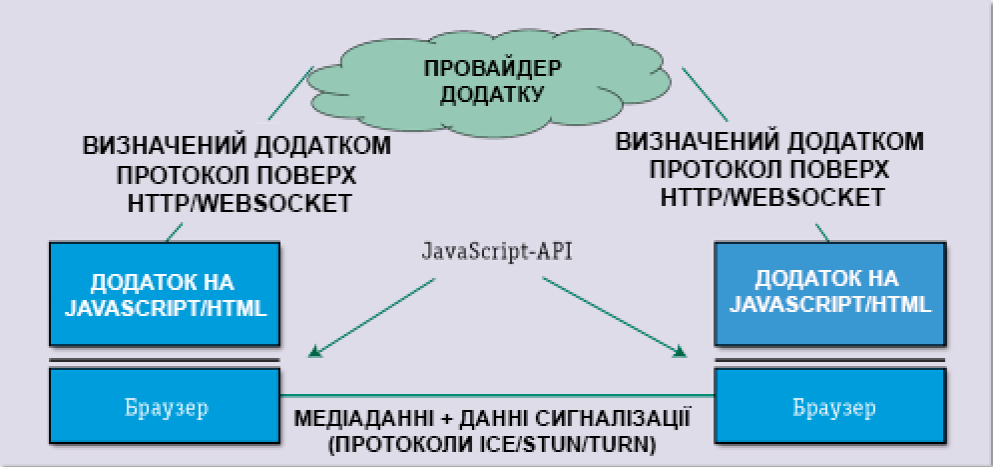


Рисунок 1.3 - Архітектура RTCWeb

Розробка такого API представляє собою цікаву задачу, але становить лише частина загальної картини, яка полягає у тому, щоб налагодити прямий зв'язок між двома або більше браузерами без посередників, що має на увазі проходження безперервних потоків даних реального часу через мережу. Таким чином, мова йде фактично про революцію в веб-комунікаціях: вперше планується забезпечити можливість однорангового зв'язків між веб-додатками, що працюють в браузерах.

Припустимо, відбувається відеорозмови по браузеру в прямому ефірі. В цьому випадку потоки відеоданих між браузерами проходять по маршруту, погодженого та організованого в рамках складної низки взаємодій за участю наступних сторін:

1. викликає браузер і викликає додаток на JavaScript (взаємодіючі через API);
2. що викликає JavaScript-додаток і провайдер додатки (як правило, веб-сервер);
3. провайдер додатки і викликається JavaScript-додаток;
4. викликається JavaScript-додаток і викликається браузер (спілкуються через той же API).

1.13 Механізм сигналізації

Загальна задача розробки WebRTC [5] полягає в тому, щоб повністю стандартизувати управління медіаданими і при цьому перенести максимальний обсяг функцій управління сигналізацією на рівень програми. Причина в тому, що різні додатки можуть віддати перевагу різні протоколи сигналізації - наприклад, SIP, XMPP або щось власне. При такому підході браузери будуть зобов'язані обмінятися наступною інформацією: опис мультімедіасеанса, яке вказує транспортний протокол і відомості для обходу NAT (по протоколу Interactive Connection Establishment, ICE), тип і формат мультимедійних файлів, а також всі інші параметри, необхідні для формування маршруту передачі.

Спочатку була запропонована ідея обмінюватися описами сеансу з використанням протоколу Session Description Protocol, але через низку непереборних проблем від цього відмовилися. У підсумку в IETF [6] зупинили вибір на JavaScript Session Establishment Protocol (JSEP), що надає інтерфейс, за допомогою якого додатки можуть працювати з локальним і віддаленим описами сеансу (узгодження якого відбувається за допомогою будь-якого механізму сигналізації), а також стандартний метод взаємодії з кінцевим автоматом ICE . Із застосуванням JSEP відповідальність за реалізацію цього кінцевого автомата несете повну додаток. Крім простий пересилання повідомлень, трансльованих від

браузера віддаленої стороні, додаток повинен викликати потрібні функції API в потрібний час, а також конвертувати опису сеансу і ICE-інформацію в повідомлення використовуваного сигнального протоколу.

1.14 Application programming interface

Використовувати нові можливості в JavaScript-додатках можна за допомогою API W3C WebRTC 1.0, який вимагає, щоб ядро браузера надавало функціональність, необхідну для організації каналів передачі аудіо-, відео- та інших даних. Але розробники стандартів ще не прийшли до рішення з приводу того, які аудіокодеки (G.711, Opus, Vorbis і т. Д.) І видеокодеки (H.264, VP8 і т. Д.) Потрібно реалізувати в браузерах. Передбачається, що медіапотоків і будь-які передачі даних завжди будуть шифруватися. API складається з трьох основних елементів: PeerConnection, MediaStreams і DataChannel.

1.15 PeerConnection

PeerConnection дозволяє встановлювати пряме з'єднання між двома браузерами і звертатися до віддаленої рівноправній стороні, зазвичай представляє собою екземпляр того ж JavaScript-додатки. Зв'язок координується по сигнальному каналу, організованому засобами скрипта на сторінці додатка (наприклад, з використанням протоколів XMLHttpRequest або WebSocket) і проходить через веб-сервер. Під час активного з'єднання, що викликає браузер може передавати об'єкти MediaStream безпосередньо віддаленого браузеру.

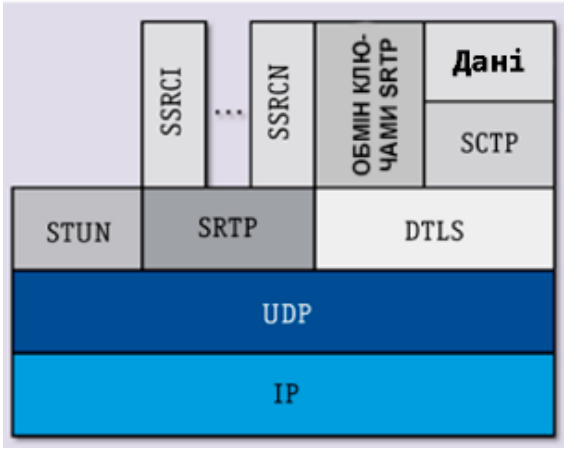


Рисунок 1.4 - Стек протоколу RTCWeb, що дозволяє захищено передавати мультиплексовані потоки по Мережі в реальному часі

Як показано на рисунку 1.4, в механізмі однорангового з'єднання використовується протокол ICE поряд з сервером Session Traversal Utilities for NAT (STUN) і протоколом Traversal Relays around NAT (TURN), щоб передані по UDP медіапотоків могли обходити засоби NAT і брандмауери. ICE дозволяє браузерам з'ясувати досить інформації про топології мережі, в якій вони використовуються, щоб організувати оптимальний комунікаційний маршрут. ICE також забезпечує необхідний рівень безпеки, не дозволяючи ненадійним веб-сторінок і додатків відправляти дані на хости, що не очікують їх отримання.

На віддалений хост кожне сигнальне повідомлення після прибуття передається приймає елементу PeerConnection. Сигнальні повідомлення, що відправляються засобами API, для більшості додатків виглядають як безглуздий набір даних, але комунікаційне веб-додаток обов'язково захищене, тому ефективніше передавати їх іншій стороні через веб-сервер.

1.16 MediaStream

MediaStream - абстрактна репрезентація реального потоку відео- або аудіо, за допомогою якої над ним можна виконувати різні дії: відображення змісту, запис або відправка віддаленого адресату. MediaStream може являти собою потік, який надходить від віддаленого вузла або відправляється на нього. Наприклад, елемент LocalMediaStream - це медіапотоків, що відправляється з локального пристрою, що реєструє (веб-камери або мікрофона).

Для створення і використання LocalMediaStream веб-додаток повинен запросити у користувача доступ за допомогою функції getUserMedia (). Додаток вказує тип мультимедійних файлів (аудіо або відео), до яких йому потрібен доступ. Селектор пристроїв в інтерфейсі браузера надає доступ або відмовляє в ньому. Після закінчення роботи додаток саме може відкликати свій власний доступ шляхом виклику функції stop () для LocalMediaStream.

Сигналізація на рівні мультимедійних файлів виконується поза каналу зв'язку між сторонами. На рисунку 1.4 показаний стек протоколів, необхідних для передачі відеоданих. За протоколом Secure Realtime Transport Protocol передаються самі медіа-дані, а по RTP Control Protocol - інформація, яка використовується для збору статистики передачі потоків. Datagram Transport Layer Security використовується для обміну ключами SRTP (Secure Realtime Transport Protocol), а також для управління асоціаціями. У IETF обговорюють можливість використання SDP Security Descriptions for Media Streams в якості

альтернативного протоколу обміну ключами і управління асоціаціями.

У мультімедіасеансе кожен вид мультимедійних файлів зазвичай передається в окремому RTP-сеансі зі своїми пакетами RTCP (RTP Control Protocol). Однак, щоб подолати проблему зі створенням нової «труби» для кожного потоку, в IETF працюють над можливістю зменшення числа портів транспортного рівня, використовуваних RTP-додатками реального часу, за рахунок об'єднання (мультиплексування) мультімедіатрафіка в єдиний RTP-сеанс.

1.17 DataChannel

DataChannel надає транспортний сервіс загального призначення, що дозволяє веб-браузерів двоспрямованістю обмінюватися даними по тимчасовій схемі. Для управління даними, що не відносяться до мультимедіа, в IETF вирішили використовувати Stream Control Transmission Protocol (SCTP), інкапсульований в DTLS (Datagram Transport Layer Security). Інкапсуляція «SCTP поверх DTLS поверх ICE поверх UDP» дозволяє обходити NAT і забезпечує конфіденційність і захист цілісності передачі. Більш того, таке рішення дозволяє транспорту даних без проблем взаємодіяти з паралельними транспортами мультимедійних файлів і ділити з ними один номер порту транспортного рівня. У IETF вибрали SCTP, оскільки він сам по собі дозволяє передавати багато потоків з надійним, ненадійним і частково надійним режимами доставки. Завдяки цьому, додатки можуть відкривати декілька незалежних потоків (до 65 536 в обох напрямках) всередині SCTP-асоціації, що пересилаються на рівноправну кінцеву точку SCTP. Кожен потік сам по собі є односпрямований логічний канал послідовної доставки.

Додаток може відправляти послідовність повідомлень впорядковано чи ні. Порядок доставки повідомлень зберігається тільки для всіх упорядкованих повідомлень, відправлених в одному потоці, проте API DataChannel є двонаправленим, а це значить, що в кожному каналі є і вхідний, і вихідний SCTP-потік.

Додаток організовує канал передачі даних (тобто створює SCTP-асоціацію), коли на об'єкті PeerConnection вперше викликається функція CreateDataChannel1 (). Кожен наступний виклик цієї функції просто створює новий канал даних в існуючій SCTP-асоціації.

1.18 Безпека та надійність технологій

RTCWeb і архітектура прямого зв'язку, без сумніву, створять нові складності в світі телекомунікацій, так як вони дозволяють веб-браузерам спілкуватися безпосередньо практично без втручання сервера. Доведеться, зокрема, розглянути цілий ряд проблем, пов'язаних з довірою і безпекою. Якщо дозволити пряму межбраузерную зв'язок, то з'являється абсолютно новий набір потенційних загроз безпеки. Нинішня базова політика безпеки Web заснована на принципі ізоляції (або «пісочниці»), який дає користувачам можливість захищати свої комп'ютери від шкідливих скриптів і підробки міжсайтових запитів. Відповідно до цієї моделі, політики безпеки в основному застосовуються до коду JavaScript, що працює в браузері, який фактично є довіреною платформою виконання відвідуваних сайтів. Коли ця платформа виходить за рамки одного браузера, відкриваються нові потенційні уразливості.

По-перше, необхідно реалізувати механізми безпеки, аналогічні наявним в інших мережевих протоколах, які забезпечують прямий зв'язок між двома будь-якими кінцевими точками (таких як SIP), - якщо не планується передбачати присутність ретрансляторів, що діють в якості прозорих для сторін посередників.

По-друге, якщо дозволити браузерам безпосередньо зв'язуватися один з одним, то повинні бути механізми, що надають користувачам можливість підтвердити свою згоду, перш ніж почати сам сеанс зв'язку. Перевірка згоди не повинна покладатися на будь-якої довірений сервер, тому її доведеться виконувати самому браузеру, який хоче почати зв'язок.

Останнє, але не менш важливе - прямий зв'язок через WWW, очевидно, потребують від браузера тісної взаємодії з пристроєм, на якому він працює. Наприклад, перш ніж зробити мультімедіавизов, необхідно отримати доступ до локальних аудіо- та відеопристроїв. Необхідно передбачити політики такого доступу, які теж зажадають отримання згоди користувача в якійсь формі, в зв'язку з чим можливий ще ряд нових проблем.

Завдання створення RTCWeb полягає в стандартизації відкритого каркаса, що дозволяє максимально просто створювати межбраузерние мультімедіапріложенія без установки будь-яких плагінів. Два найбільших розробника браузерів вже випустили ранні реалізації такого фреймворка, проте жодна з них поки повністю не відповідає стандарту, хоча обіцяно, що скоро це зміниться.

У найближчому майбутньому робочі групи по стандартизації зв'язку реального часу в Web займуться контролем заторів, вибором аудіо- та відеокодеків і оптимізацією

використання каналів передачі даних.

1.19 Постановка наукової задачі та обґрунтування методики досліджень

В результаті огляду існуючих протоколів та технологій було визначено основні рішення для кросплатформного обміну мультимедійним контентом.

Задачі магістерської роботи:

1. Дослідження, аналіз та порівняння обраних технологій для передачі мультимедійного контенту.
2. Удосконалення кращої технології.
3. Розробка і тестування технології для передачі мультимедійного контенту.

1.20 Висновки до розділу 1

У першому розділі магістерської роботи проведено огляд існуючих технологій для передачі інформації між двома вузлами мережі. Були розглянуті як однорангові, так і багаторангові мережі.

Було визначено деякі проблеми передачі мультимедійного контенту.

Під час досліджень, проведених в першому розділі даної роботи, були виділені переваги та недоліки різних протоколів та технологій для передачі інформації між пристроями на різних платформах.

РОЗДІЛ 2
ОСНОВНІ ТЕХНОЛОГІЇ ДЛЯ ПЕРЕДАЧІ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

У даному розділі проведено аналіз, дослідження та порівняння трьох основних технологій для передачі контенту.

2.1 Існуючі технології для передачі мультимедійного контенту

2.1.1 Java аплети

Часом появи браузерних дзвінків можна вважати момент, коли Java аплети стали підтримувати захоплення аудіо з мікрофона. Java Runtime Environment (JRE) [7], як правило, вже встановлена в системі, будь-то Linux або Windows. JRE так само присутній у вигляді плагіна в більшості відомих браузерів. Наприклад, якщо заглянути у вкладку chrome: // plugins браузера Google Chrome, можна знайти там NPAPI Java плагін. Цей плагін і буде середовищем для виконання Java аплету [8,9]. Другий, більш просунутий спосіб запуску, це JNLP (Java Network Launch Protocol) [10,11].

В результаті Java код виконується десктопом або браузерним плагіном, захоплює аудіо з мікрофона і відправляє його на сервер по стандартному RTP протоколу. З безпекою тут так само все в порядку. Зрозуміло, що аплет повинен бути підписаний, і при запуску користувача спитають, чи бажає він запустити підписаний аплет від даного виробника, який може отримати доступ до тих або інших функцій.

Плюси рішення на Java:

1. простота;
2. немає зайвих ланок, можливість прямої взаємодії з сервером по RTP;
3. широка поширеність і доступність JRE для кінцевого користувача.

Мінуси:

1. відсутність AEC (Acoustic Echo Cancellation);
2. відсутність AGC (Automatic Gain Control);
3. відсутність Adaptive Jitter Buffer;
4. відсутність Noise suppression.

Відсутність AGC змусить ваших користувачів крутити рівень гучності (нормальний AGC повинен робити це за користувача, щоб не було занадто тихо або дуже голосно). А

відсутність Adaptive Jitter Buffer виллється або в більшу затримку або в «choppy audio» - переривчастий нерозбірливий звук. В результаті якість комунікації буде далека від оптимальної.

Всі відсутні алгоритми можна теоретично реалізувати на Java, але є пара проблем. По-перше, реалізувати універсальні і продуктивні алгоритми (наприклад AEC) досить складно: така реалізація вимагатиме високих трудозатрат і витрат. По-друге, реалізація таких алгоритмів на Java може працювати в кілька разів повільніше, ніж на C / C ++, що може спричинити серйозні проблеми з продуктивністю і перевитрата ресурсів клієнтського CPU.

Виробники Java аплетів з функцією дзвінків реалізують власні DSP процесори або використовують вже існуючі рішення на C / C ++. Як правило, вони підкладають до аплету DLL бібліотеки, які беруть на себе обробку вищеописаних DSP алгоритмів. В результаті Java аплет має стандартні VoIP функції для забезпечення якісного дзвінка з усіма VoIP алгоритмами.

В кінцевому підсумку залишається два мінуси Java:

кроссплатформенность - доведеться забезпечувати аплети DLL бібліотеками під Win і SO бібліотеками під Linux;

складність реалізації цих DSP бібліотек.

Довести DSP до відмінної якості або купити відповідні розробки може дозволити собі не кожен вендор. Те ж стосується підтримки різних аудіо- і відеокодеків.

2.1.2 Flash player та adobe media server

До деякого часу Flash був технологією для красивих інтерактивних банерів. У 2002 році з'явилася в релізі версія Flash Communication Server MX 1.0 - перша версія сьогоднішнього Adobe Media Server [12,13]. Flash Player 6, будучи тоді продуктом компанії Macromedia, вмів взаємодіяти з FCS MX 1.0 і обмінюватися з сервером аудіопотоками.

В цей час Flash Player 6 вже вмів захоплювати аудіо і жати його в NellyMoser і відео - в Sorenson Spark. Java в цей час була слабо представлена в веб та Flash Player 6 в зв'язці могли покривати всі потреби користувачів. В якості транспорту для аудіо та відео в Flash Player 6 використовувався протокол RTMP [14], який сьогодні має відкриту специфікацію, яка опублікована Adobe. Flash Player 6 в зв'язці з сервером став платформою для браузерних дзвінків, що володіє наступними функціями:

1. NellyMoser, Sorenson spark;

2. RTMP [15,16].

Основною проблемою була затримка між користувачами. Затримка була у зв'язці Flash Player 6 + Flash Communication Server MX 1.0, а також залишилася в наступних версіях сервера, включаючи останню Adobe Media Server. Причина в тому, що RTMP протокол працює поверх TCP, а тому не пристосований для повноцінного VoIP.

Відсутність повноцінного UDP транспорту унеможливив розвиток інтерактивних сервісів. Наприклад, якщо учасник вебінару хоче поспілкуватися face-to-face з глядачем, у нього не завжди вийде це зробити нормально. Все в значній мірі залежить від якості мережі і втрат. В результаті на базі RTMP розвивалися, в основному, video on demand сервіси, або live video, або вебінари з одним ведучим, де затримка не так важлива.

Ситуацію з UDP в Flash Player зрушила компанія Adobe, коли у версії плеєра 10 ввели підтримку нового протоколу RTMFP і ехоподавлення. У 11 версії Flash Player додали підтримку кодеків G.711 і H.264. У AS3 API так само є згадки про Adaptive Jitter Buffer для G.711 і Speex.

VoIP можливості Flash Player 32:

1. Nelly Moser, Speex, G.711, Sorenson Spark, H.264;
2. RTMP, RTMFP;
3. AEC;
4. Adaptive Jitter Buffer;
5. AES шифрування.

Завдяки цьому, Flash Player має практично все необхідне, щоб стати VoIP платформою №1 для браузера: шифрування AES захищає трафік між браузером і сервером від сторонніх; AEC і Jitter Buffer забезпечують якість відтворення звуку; нові кодеки сумісні з традиційним VoIP, а RTMFP протокол працює по UDP. Бракує AGC (Automatic Gain Control. Ще одна маленька неприємність від Adobe - неможливість у Flash Player нормально програти H.264 потік, закодований для передачі по RTP. Це обмеження вимикає всі H.264 кодеки, які дають потік, сумісний з RTP. В результаті доводиться застосовувати транскодинг, що в свою чергу викликає надмірне споживання ресурсів CPU, якого в принципі хотілося б уникнути.

Flash RTMFP заснований на UDP і досить пристойно передає звук. Але і тут не обійшлося без проблем. У інструкціях Adobe AS3 сказано, що RTMFP для аудіо та відео підтримує три режими: надійна доставка (reliable), частково-надійна доставка (partially reliable), ненадійна доставка (unreliable). Але є тільки два прапора для audioReliable і videoReliable: true, false. False описується як режим часткової доставки (partially reliable). У

підсумку, виходить, що ненадійна доставка (unreliable) пропала, а для передачі звуку вона найбільш важлива. Часткова доставка - це TCP подібні ретрансміти, які відбуваються дуже обмежений час, але і цього вистачає, щоб зіпсувати звук на нестабільній мережі. Такі ретрансміти викликають джиттер, який псує потік. Jitter buffer на приймаючій стороні в даному випадку не допомагає, тому що йде дуже великий розкид. Рішенням є перехід до ненадійної доставки (unreliable) звуку. У Flash Player API немає можливостей її включити, і доводиться додавати її на рівні протоколу на серверній стороні.

Плюси Flash:

1. звична технологія для розробників - Flex / AS3;
2. якісна VoIP передача аудіо і відео.

Мінуси Flash:

1. вимагає проміжного сервера (не підтримує відкриті UDP протоколи, такі як RTP / SRTP);
2. відсутність AGC, H.264.

2.1.3 Web Real Time Communications

Абревіатура WebRTC [17] розшифровується як Web Real Time Communication (вебкоммунікація в режимі реального часу) - це відкритий стандарт для впровадження можливостей мультимедійного зв'язку в реальному часі безпосередньо в веб-браузері. Використання платформи, заснованої на відкритому стандарті, дозволяє відмовитися від завантаження додаткових програм, надбудов і розширень.

WebRTC відрізняється не тільки простотою застосування. Можна користуватися послугою WebRTC не потрібно багато ресурсів, оскільки сервер з'єднує тільки співрозмовників. установка з'єднання також не представляє особливої складності. Спочатку браузер подає сервера WebRTC сигнал, що він планує розпочати розмову далі він отримує HTTPS-посилання і зв'язок здійснюється в зашифрованому вигляді. Після цього браузер запитує у користувача дозвіл на доступ до веб-камері і мікрофону. Цей метод дозволяє встановити пряме з'єднання між двома браузерами таким чином, що аудіо- та відеодані не проходять через сервер, де може статися перевантаження. Завдяки прямому з'єднанню для WebRTC не потрібна ні реєстрація, ні обліковий запис в будь-якій службі. Для початку бесіди потрібно тільки пройти по посиланню. Спілкування залишається приватним, оскільки потік даних шифрується .

Спілкування по технології WebRTC [18] в браузері запускається за допомогою коду

JavaScript. після цього за комунікацію відповідають три частини: голосова і відео частини збирають мультимедійні дані з веб-камери і мікрофона, а транспортна частина об'єднує інформацію і пересилає потік в зашифрованому вигляді, використовуючи протокол SRTP.

WebRTC грунтується на продукті від компанії Global IP Solution (GIPS), яка була куплена компанією Google в травні 2010-го. Комунікацією в реальному часі через браузер Google почала активно займатися в 2011 році. У той же рік вони опублікували вихідний код своєї реалізації WebRTC. Технологія використовує свої аудіокодеки і відкритий формат відео VP8.

Для компаній використання технології WebRTC відкриває можливості трансформувати зв'язок, надаючи надійні та безпечні комунікації корпоративного класу, тим самим відкриваючи можливості для організацій онлайн-нарад, відеоконференцій та інших заходів.

Безсумнівно, у даній технології є ряд важливих переваг, які виводять цю технологію на високий рівень:

1. економія часу користувачів на установку і підтримку розширень або плагінів, а також легке підключення до відеоконференції;
2. WebRTC - забезпечує більш високий рівень безпеки, ніж більшість сучасних систем телефонного зв'язку;
3. проект з відкритим кодом - легко впровадити в свій стартап;
4. новий рівень онлайн підтримки - спілкуйтеся з користувачем прямо зі сторінки вашого корпоративного сайту.

WebRTC має більш високу якість звуку, в порівнянні з плагінами на Flash, для передачі аудіо використовуються кодеки Opus і G.711, а з точки зору стабільності браузера і захищеності по відношенню до зовнішніх атак WebRTC випереджає будь-які плагіни, що є безперечним плюсом .

Таким чином, існуючі проблеми та недоліки WebRTC вирішуються технологія

активно підтримується і розвивається в першу чергу компанією Google. Вона впроваджена в такі браузери як Google Chrome, Mozilla Firefox, Opera та інші. Є підтримка Web Real-Time Communication для мобільних пристроїв Android.

На технологіях WebRTC хотілося б зупинитися окремо, це: SRTP, DTLS, ICE, STUN, AEC, AGC, Adaptive Jitter Buffer, Opus, VP8

Набір використовуваних в WebRTC технологій більше схожий на VoIP SDK. SRTP і DTLS забезпечують захист трафіку між WebRTC вузлами. ICE і STUN допомагають подолати NAT, виставивши по обидва боки кандидатів для додзвону у вигляді простих пар

host: port. AEC, AGC і Jitter Buffer працюють для того щоб зробити аудіо і відео якісним - без лагів і затримок. Кодеки Opus і VP8 добре підходять для глобального Інтернету, де бітрейт до кінцевого користувача може легко падати до дуже низьких значень всупереч обіцянкам провайдерів про канали в 100Mbps.

WebRTC спочатку задумана як peer-to-peer між браузерами і орієнтована на шифрований SRTP трафік. Швидше за все, саме тому VoIP вендорам, які обмінюються стандартними RTP потоками, доведеться впроваджувати у себе WebRTC сумісний транспорт і кодеки.

Хоча і тут не все так погано. Існують шлюзи для забезпечення такого роду сумісності. Наприклад WebRTC SIP Gateway Flashphoner Web Call Server 3 є шлюзом, який може з'єднати WebRTC клієнта і стандартне SIP / RTP пристрій будь то VoIP світч або GSM шлюз, який працює по SIP. Таким чином, дана проблема цілком вирішувана шляхом введення проміжного програмного забезпечення.

Переваги та недоліки WebRTC досить прозорі.

Плюси WebRTC:

1. повноцінне VoIP в браузері;
2. підтримує бульшість відомих браузерів.

Мінуси WebRTC:

1. відсутність сумісності з традиційними VoIP;
2. не підтримує IE.

2.1.4 Схема роботи Java (JMF)

Можно відтворювати вхідні потоки RTP локально, зберігати їх у файл або використовувати обидва варіанти (рис. 2.1).

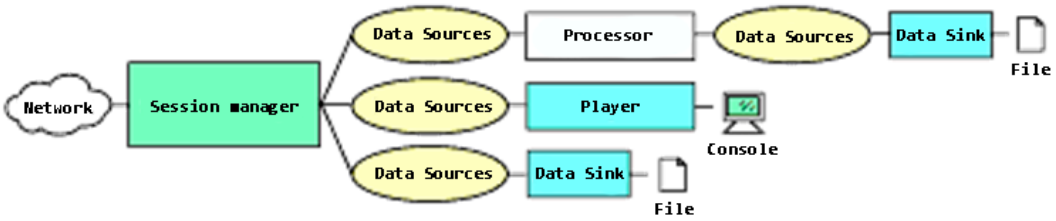


Рисунок 2.1- Прийом RTP

Можно використовувати API RTP для передачі захоплених або збережених медіапотоків по всій мережі. Вихідні потоки RTP можуть походити з файлу або пристрою

захоплення. Вихідні потоки також можна відтворити локально, зберегти у файл або обидва (рис. 2.2).

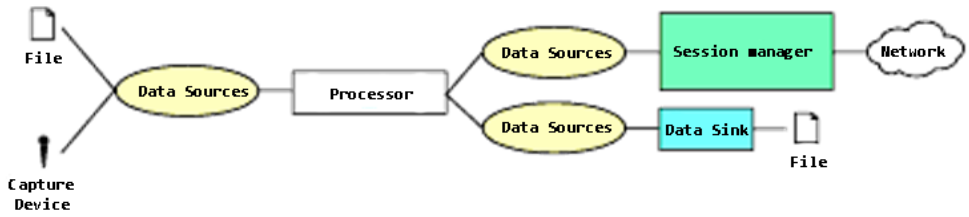
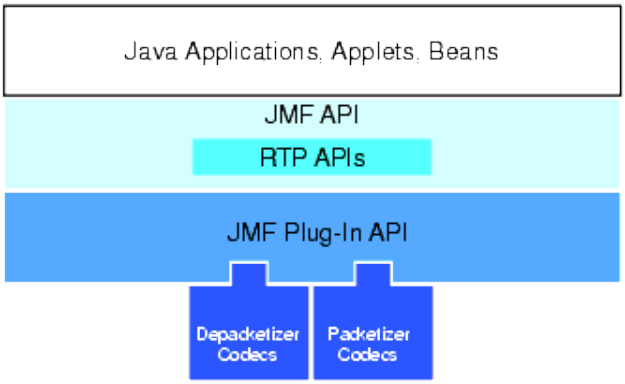


Рисунок 2.2 - Передача RTP

API RM JMF [11,12] розроблений так, щоб легко працювати з можливостями збору, представлення та обробки JMF. Програвачі та процесори використовуються для представлення та маніпулювання потоками медіа RTP, як і будь-який інший медіа-контент.

Для передачі медіа потоків, які були захоплені з локального пристрою захоплення за допомогою DataSource захоплення або які були збережені у файл за допомогою DataSink (рис. 2.3).



Риснок 2.3 - Архітектура RTP високого рівня JMF.

У JMF для координації сеансу RTP використовується SessionManager. Він відстежує учасників сесії та потоки, які передаються. Він обробляє канал управління RTCP і підтримує RTCP як для відправників, так і для приймачів.

Інтерфейс SessionManager визначає методи, які дозволяють програмі ініціалізуватись та починати участь у сеансі, видаляти окремі потоки, створені програмою, та закривати весь сеанс. Менеджер сеансу підтримує статистику всіх пакетів RTP і RTCP, надісланих і отриманих у сеансі. Статистика відслідковується протягом усього сеансу на основі потоку.

Менеджер сесії веде облік усіх учасників сесії. Кожен учасник представлений екземпляром класу, який реалізує інтерфейс учасника. SessionManager створює Учасника щоразу, коли надходить пакет RTCP, який містить опис джерела (SDES) з канонічним іменем (CNAME). Учасник може володіти більш ніж одним потоком, кожен з яких ідентифікується за допомогою ідентифікатора джерела синхронізації (SSRC), використовуваного джерелом потоку. SessionManager підтримує об'єкт RTPStream для кожного потоку пакетів даних RTP в сеансі. Існує два типи потоків RTP: ReceiveStream являє собою потік, який надходить від віддаленого учасника.

SendStream являє собою потік даних, що надходять від Процесора або вхідних джерел даних, що надсилаються по мережі.

ReceiveStream створюється автоматично, коли менеджер сеансів виявляє нове джерело даних RTP.

У javax.media.rtp.event визначено декілька специфічних для RTP подій. Ці події використовуються для повідомлення про стан сеансу RTP та потоки (рис. 2.3).

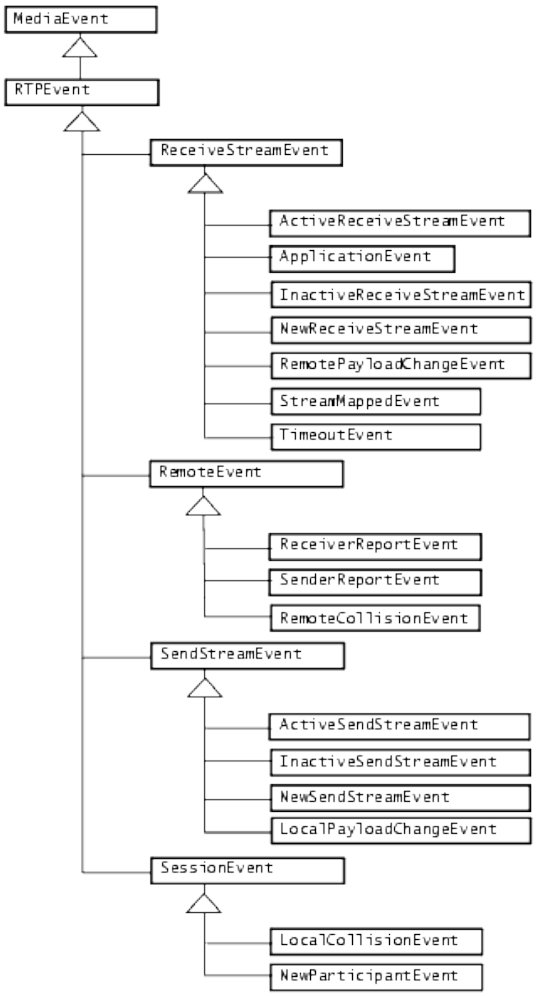


Рисунок 2.3 - RTP події

Щоб отримувати сповіщення про події RTP, потрібно впровадити відповідного слухача RTP та реєструєте його у менеджера сеансу:

1. SessionListener : отримує повідомлення про зміни стану сеансу;
2. SendStreamListener : отримує сповіщення про зміни стану потоку RTP, який передається;
3. ReceiveStreamListener : отримує сповіщення про зміни стану потоку RTP, що надходить;
4. RemoteListener : отримує сповіщення про події або повідомлення управління RTP, отримані від віддаленого учасника;
5. NewParticipantEvent : вказує, що до сеансу приєднався новий учасник;
6. LocalCollisionEvent : вказує, що джерело синхронізації учасника вже використовується.

Існує п'ять типів подій, пов’язаних із SendStream :

1. NewSendStreamEvent : Вказує, що місцевий учасник створив новий потік відправки;
2. ActiveSendStreamEvent : вказує на те, що розпочато передачу даних із джерела даних, використовуваного для створення потоку надсилання;
3. InactiveSendStreamEvent : Вказує на те, що передача даних із DataSource, що використовується для створення потоку відправки, зупинена;
4. LocalPayloadChangeEvent : вказує на те, що формат потоку або корисне навантаження змінилися;
5. StreamClosedEvent : вказує на те, що потік закрито.

Існує сім видів подій, пов’язаних із ReceiveStream :

1. NewReceiveStreamEvent : вказує, що менеджер сеансу створив новий потік прийому для щойно виявленого джерела;
2. ActiveReceiveStreamEvent : вказує на те, що передача даних розпочалася;
3. InactiveReceiveStreamEvent : вказує на те, що передача даних припинена;
4. TimeoutEvent : Позначає, що передача даних закінчилася;
5. RemotePayloadChangeEvent : вказує на те, що формат або корисна навантаження потоку прийому змінилися;
6. StreamMappedEvent : вказує на те, що раніше осиротілий потік прийому асоціювався з учасником;
7. ApplicationEvent : вказує на те, що отримано пакет додатків RTCP APP.

Існує три типи подій, пов'язаних з віддаленим учасником:

1. ReceiverReportEvent : позначає, що отримано звіт про приймач RTP;
2. SenderReportEvent : позначає, що отримано звіт про відправника RTP;
3. RemoteCollisionEvent : вказує, що два віддалені учасники використовують один ідентичний ідентифікатор джерела синхронізації (SSRC).

Потоки в межах сеансу RTP представлені об'єктами RTPStream . Існує два типи RTPStreams : ReceiveStream і SendStream . Кожен потік RTP має пов'язане з ним джерело даних буфера. Для ReceiveStreams цей джерело даних завжди є PushBufferDataSource .

Всі дані, що стосуються RTP, використовують кодування RTP-специфічного формату, як визначено в класах AudioFormat та VideoFormat .

AudioFormat визначає чотири стандартних RTP-специфічних рядка кодування:

1. статичний кінцевий рядок ULAW_RTP = "JAUDIO_G711_ULAW / rtp";
2. статичний кінцевий рядок DVI_RTP = "dvi / rtp";
3. статичний кінцевий рядок G723_RTP = "g723 / rtp";
4. статичний кінцевий рядок GSM_RTP = "gsm / rtp".

VideoFormat визначає три стандартні RTP-специфічні рядки кодування:

1. статичний кінцевий рядок JPEG_RTP = "jpeg / rtp";
2. статичний кінцевий рядок H261_RTP = "h261 / rtp";
3. статичний кінцевий рядок H263_RTP = "h263 / rtp".

RTPControl надає методи доступу до статистики сеансів та отримання поточного формату корисного навантаження.

Подання вхідного потоку RTP здійснюється програвачем . Щоб отримати та представити один потік із сеансу RTP, можна використовувати MediaLocator, який описує сеанс, щоб побудувати програвач . Медіа-локатор для сеансу RTP має форму:

rtp: // адреса: порт [: ssrc] / content-type / [ttl]

Якщо в сеансі є кілька потоків, які треба представити, потрібно використовувати менеджер сеансів .Можна отримувати сповіщення від менеджера сеансу кожного разу, коли в сеанс додається потік та створюється програвач для кожного нового потоку. Використання менеджера сеансу також дозволяє безпосередньо контролювати сеанс та контролювати його. Щоб створити потік передачі для передачі даних з живого джерела захоплення:

1. створити, ініціалізувати та запустіти SessionManager для сеансу;
2. створити процесор за допомогою відповідного DataSource захоплення;
3. встановити вихідний формат процесора у формат, характерний для RTP;

4. отримати вихідне джерело даних з процесора;
5. отримати createSendStream у менеджері сеансів та перейти у DataSource.

2.1.5 Схема роботи WebRTC

Загальна архітектура WebRTC представлена на рисунку 2.4.

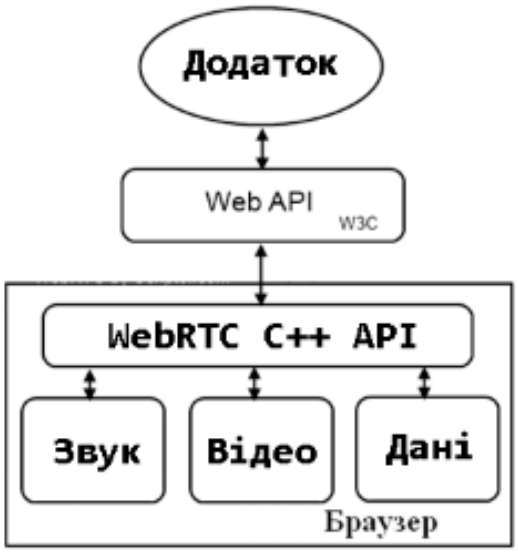


Рисунок 2.4 - Загальна архітектура WebRTC

Основною частиною стека WebRTC [19,20] є C ++ API - набір бібліотек і інтерфейсів, написаних на C ++. Передбачається, що розробники різних web-клієнтів або браузерів можуть вбудувати в свої додатки набір C ++ компонентів WebRTC і забезпечити до них доступ через високорівневе Web API на JavaScript. Цей набір бібліотек забезпечує весь функціонал стека WebRTC:

Транспорт - компоненти для організації підключення і управління призначеними для користувача сесіями мають функціонал для обходу NAT, проксі і файерволів шляхом попереднього узгодження з'єднання через STAN і TURN сервера, передачі потоку по протоколах RTP або SRTP, які працюють поверх TCP і UDP, шифрування потоку при передачі по SRTP, управління смугою пропускання.

Движок обробки аудіо - компонент, що забезпечує передачу аудіосигналу від аудіокарти до мережевого інтерфейсу і назад. Він включає в себе аудіо кодеки iSAC і Opus, системи ехоподавлення і придушення шуму. Кодек iSAC є широкосмуговим аудіо кодеком

для VoIP і потокового аудіо, використовує частоту дискретизації 16/32 кГц з адаптивною передачі даних від 12 до 52 Кбіт / с. Opus підтримує постійний і змінний бітрейт від 6 до 510 Кбіт / с і частоти дискретизації від 8 до 48 кГц.

Движок обробки відео - компонент для управління видеосигналом, що включає кодек VP8, систему адаптивного відеобуффера Jitter Buffer, автоматично підлаштовуватися під поточний стан з'єднання і систему корекції зображення. Кодек VP8 має підвищену стійкість до втрати пакетів, механізм фільтрації артефактів, який може застосовуватися по-різному до різних частин кадру в залежності від їх зміни, можливість масштабувати декодування, має профілі, оптимізовані для проведення відео-конференцій в реальному часі.

C++ API має два основних інтерфейсу: Stream API і PeerConnection, які, в свою чергу, поділяються на кілька дочірніх інтерфейсів. Stream API призначений для отримання медіапотоку від апаратного забезпечення клієнта (веб-камера / мікрофон) або перетворення в потік даних, які надасть користувач, наприклад, для мовлення заготовленого користувачем файлу і обробки відео та аудіо потоків. PeerConnection API призначений для узгодження і створення з'єднання, обходу NAT, брандмауера або проксі, і передачі потоку по мережі. Загальна схема роботи представлена на рисунку 2.5.

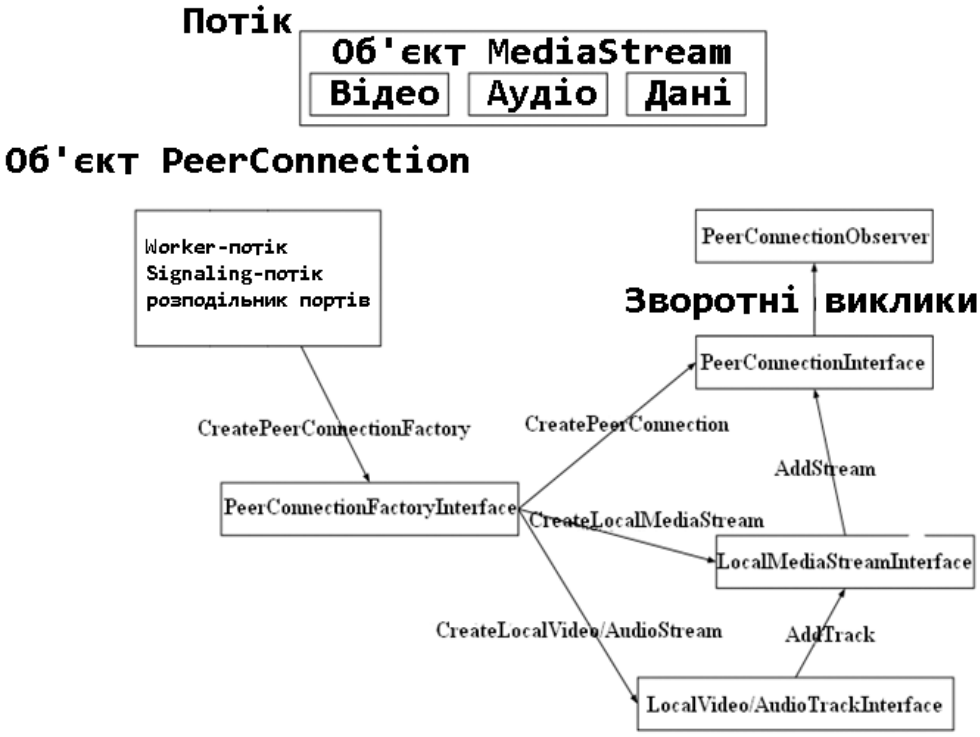


Рисунок 2.5 - Схема взаємодії з C ++ API

Розробники додатків можуть впровадити в свої додатки C ++ API і забезпечити до нього доступ через високорівневе Web API на JavaScript, яке описується стандартом W3C. Розробники кінцевих клієнтських додатків можуть використовувати Web API для доступу до функцій WebRTC, не вдаючись у подробиці внутрішньої архітектури WebRTC і концентруючи увагу на архітектурі свого web-додатку. Web API, також як і C ++ API, має два основних інтерфейсу: MediaStream і RTCPeerConnection.

Однією з цікавих особливостей MediaStream, є те, як він працює з аудіо і відео, приймаючи дані від кожного пристрою і передаючи його у вигляді потоку. При отриманні аудіо і відео MediaStream працює окремо з кожним потоком. Таким чином, є можливість відправляти або отримувати від інших клієнтів відразу кілька незалежних потоків. Це особливо актуально для портативних пристроїв, таких як планшети і смартфони, які можуть мати більше однієї web-камери (задня і фронтальна) або мікрофона. MediaStream не обов'язково повинен бути відправлені через Інтернет. Потік може бути використаний локально, перетворений, сконвертовані або ж записаний в файл.

RTCPeerConnection займається установкою прямого підключення і управляє потоками між клієнтами. Він може бути асоційований з двома наборами потоків: вхідних і вихідних. Вихідний потік відправляється в об'єкт RTCPeerConnection від MediaStream і мовить іншим клієнтам, що входить потік передається назад в MediaStream для обробки і відображення в інтерфейсі, якщо це необхідно.

2.1.6 Схема роботи Flash

Загальна архітектура на рисунку 2.6

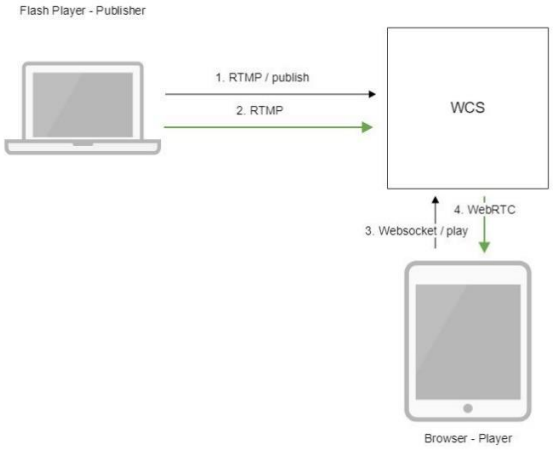


Рисунок 2.6 – Архітектура Flash

Flash Player з'єднується з сервером по протоколу RTMP і відправляє команду publish.

Flash Player захоплює мікрофон і камеру і відправляє RTMP потік на сервер.

Браузер встановлює з'єднання з Websocket і відправляє команду play.

Браузер отримує WebRTC потік і відтворює цей потік на сторінці.

На рисунку 2.7 показана послідовність викликів при використанні прикладу Flash Streaming.

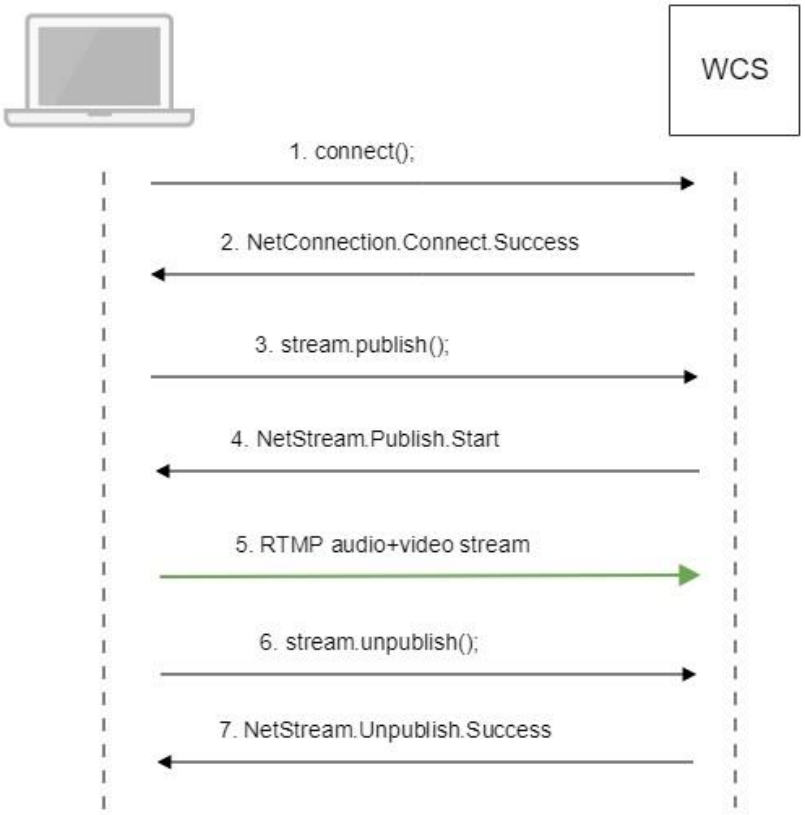


Рисунок 2.7 – Послідовність викликів

2.1.7 Фінальне порівняння

Таблиця 2.1 - Системні вимоги

Технологія	RTMP (FLASH)	JMF (RTP)	WebRTC
Операційна система	Linux CentOS 6	Solaris/Linux	Linux Debian 8, 9
Хмарні рішення	AWS, Google, Azure	-	AWS, Google, Azure
Версія	1.3.0 EOL	2.1.1e	1.0
CPU	> 2 ядра OpenGL	> 2 ядра	> 2 ядра
RAM	> 2 Gb	> 2 Gb	> 4 Gb
HDD/SDD	> 4 Gb	> 10 Gb	> 10 Gb

Таблиця 2.2 - Основні характеристики

Технологія	RTMP (FLASH)	JMF (RTP)	WebRTC
SPLIT модуль	Так	Ні	Так
Відеоканали на одиницю	1-120	1-10	1-240
Web Plugin	Flash Player	Java	Не потребується
SIP Connect	Version 11	Version 4.6	Version 13
Пріоритет голосу над відео	Так	-	Так
Mobile SDK	Не підтримується	Android	iOS & Android
API Framework	Flash SWF Editor	Java SE	VideoRTC.js
Веб панель	Не підтримується	Для дебагу та моніторингу	Для дебагу та моніторингу

Таблиця 2.3 - Протоколи зв'язку

Технологія	RTMP (FLASH)	JMF (RTP)	WebRTC
Протокол	Adobe’s Open Public Protocol	RTSP	ETF Open Standard Protocol
Аудіо кодек	G711, Speex	G711, A-law, IMA4 ADPCM	G711, Opus
Відео кодек	H263 Sorenson	Cinepak, MJPEG, RGB, YUV, VCM, H.261, H.263	VP8, VP9, H264
IP-пакет	TCP	UDP	UDP
Формат експорту відео	Не підтримується	.mpeg2 .mpeg4	.mweb .mp4
Автовиклик	Так	Ні	Так
Автологін	Так	Так	Так

Таблиця 2.4 - Підтримка веб-браузерів

Технологія	RTMP (FLASH)	JMF (RTP)	WebRTC
Chrome	FLASH заблоковано	Заблоковано	Підтримується
Firefox	FLASH заблоковано	Заблоковано	Підтримується
MS Internet Explorer	Підтримується	Заблоковано	Не підтримується
MS Edge	FLASH заблоковано	Заблоковано	Не підтримується з 03-2019
Opera	Підтримується	Заблоковано	Підтримується
Safari	FLASH заблоковано	Заблоковано	Тільки звук

Таблиця 2.4 - Спеціальні можливості

Технологія	RTMP (FLASH)	Java (RTP)	WebRTC
Передача файлів	Не підтримується	Підтримується	Підтримується
Захват екрану	Не підтримується	Підтримується	Підтримується
Скріншот	Не підтримується	Підтримується	Підтримується
Запис відео	Не підтримується	Підтримується	Підтримується
Спільний браузер	Не підтримується	Не підтримується	Підтримується

2.2 Використання Web API

2.2.1 Створення підключення

Для того, щоб створити об'єкт WebRTC-підключення, необхідно викликати конструктор RTCPeerConnection, який приймає два аргументи в якості параметрів: об'єкт зі списком ICE серверів і об'єкт з параметрами підключення. Конструктор поверне об'єкт підключення, на якому будуть викликатися всі інші описані нижче методи.

Незважаючи на те, що WebRTC дозволяє реалізувати пряму передачу даних між браузерами, необхідно використовувати проміжні сервера для координації з'єднання, обходу NAT і файерволов. Signaling, процес координації підключення, необхідний для того, щоб браузери перед початком передачі даних могли обмінятися службовою інформацією. Ця службова інформація передається від клієнта до клієнта в форматі SDP і може містити:

1. повідомленнями управління сесіями, які необхідні для відкриття або закриття сеансу зв'язку;
2. повідомленнями про помилки;
3. метаданими, такими як інформація про кодеки, типу переданих даних і т.д.;
4. ключовими даними для установки захищеного з'єднання;
5. мережевий інформацією, такий як IP-адреса і порт.

Приклад SDP при координації WebRTC-з'єднання для передачі аудіо та відео показаний на рисунку 2.8.

```
v=0
o=- 1758966045304012812 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS mcjw5hlyaawO3UAD5eCoRMbuVjJJPAKuspWv
m=audio 1 RTP/SAVPF 111 103 104 0 8 106 105 13 126
c=IN IP4 0.0.0.0
a=rtcp:1 IN IP4 0.0.0.0
a=ice-ufrag:rQ0nRK34upAOZ5Mq
a=ice-pwd:wpxbDUvRlBXHSxGhGzeoDiez
a=ice-options:google-ice
a=fingerprint:sha-256 24:35:C4:8A:13:51:B9:C6:EB:C4:2F:01:A1:BA:91:7F:90:DB:DD:64:7C:49:F9:66:0A:2C:62:B1:93:6B:05:F3
a=setup:actpass
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

Рисунок 2.8 - Приклад SDP-повідомлення

SDP генерується методами об'єкта RTCPeerConnection createOffer і createAnswer. Перший створює SDP для запиту підключення, другий - SDP для

відповіді. Коли клієнт отримує від іншого клієнта запит на підключення, він може відправити відповідь, згенерований createAnswer. Після того як обидва клієнта збережуть одержані одна від одної SDP методом setRemoteDescription об'єкта RTCPeerConnection, а свої власні SDP збережуть методом setLocalDescription, з'єднання між цими клієнтами буде створено. Процес відправки і прийняття SDP-повідомлень може бути реалізований за допомогою серверних сокетів або баз даних реального часу.

Більшість користувачів в Інтернет захищені кількома шарами NAT, можуть мати антивірусне програмне забезпечення, що блокує певні порти і протоколи або виходять через проксі і фаєрволи і тому не можуть обмінюватися інформацією. Файервол і NAT може бути реалізований одним і тим же пристроєм, таким як звичайний Wi-Fi роутер. WebRTC додатки можуть використовувати ICE, щоб обійти обмеження, накладені NAT і файерволами. Для цього необхідно задати об'єкту RTCPeerConnection список STUN і TURN серверів, як це показано на рисунку 2.9.

```
202 var ICEServers = {
203     iceServers: [
204         { url: 'stun:stun4.lgoogle.com:19302' },
205         { url: 'turn:turn.anyfirewall.com:3478?transport=udp',
206             username: 'webrtckamtest',
207             credential: 'kfSDjHJ' },
208         { url: 'turn:turn.anyfirewall.com:443?transport=tcp',
209             username: 'webrtckamtest',
210             credential : 'kfSDjHJ' }
211     ]
212 };
213 var Peer = new RTCPeerConnection(ICEServers, optional)
```

Рисунок 2.9 - Створення RTCPeerConnection зі списком ICE-серверів

ICE намагатиметься знайти найкращий шлях для підключення. Він спробує всі можливості і вибирає найбільш ефективний варіант. ICE спочатку спробує встановити з'єднання з використанням адреси, отриманого від операційної системи і мережевої карти. Якщо це не вдається (клієнт знаходиться за одним або кількома NAT), ICE отримує зовнішній адресу, використовуючи сервер STUN, і якщо це також не вдається, трафік прямує через TURN сервера. Іншими словами:

STUN сервер використовується для отримання зовнішнього мережевого адреси клієнта, що знаходиться за NAT;

TURN сервер використовуються для передачі трафіку, якщо пряме з'єднання втрачено

2.2.2 Отримання локального потоку

Отримання доступу і отримання медіа потоків від користувача реалізується засобами, які надає MediaStream API. Основним методом MediaStream - метод getUserMedia, що викликається на глобальному об'єкті navigator . getUserMedia приймає в якості параметрів три аргументи. Перший вказує на те, які дані необхідно отримати - аудіо, відео або обидва. Другий аргумент - функція, яка буде викликана при вдалому отриманні потоку, аргументом цієї функції буде об'єкт потоку. Третій аргумент - функція, яка буде викликана при виникненні помилки або в тому випадку, якщо користувач заборонить взяття медіа потоків. Після виклику getUserMedia браузер запросить у користувача дозвіл на взяття потоку. Ім'я методу getUserMedia відрізняється в різних браузерах і викликається з префіксами, тому що WebRTC ще не до кінця стандартизований. Так, в Mozilla Firefox, getUserMedia викликається з префіксом moz, в Chrome - з префіксом webkit . Простий приклад виклику getUserMedia для отримання аудіо і відео на рисунку 2.10.

```
217 var constraints = { audio: true, video: true };
218
219 function success(stream) {
220     //Поток получен
221 }
222
223 function error() {
224     //Ошибка
225 }
226
227 navigator.webkitGetUserMedia(constraints, success, error);
228
```

Рисунок 2.10 - Виклик getUserMedia

Після того, як локальний потік отриманий, він може бути показаний користувачеві. Це реалізується засобами HTML5. Відео може бути відображено на web-сторінці шляхом додавання HTML-елемента video, в який доданий потік через властивість src як показано на рисунку 2.11.

```
229
230     video.src = window.webkitURL.createObjectURL(stream);
231
```

Рисунок 2.11 - Приєднання потоку

Після того, як потік користувача отриманий і відображений, він може бути відправлений іншому користувачеві.

2.2.3 Відправка та отримання потоків

Для відправки локального потоку, отриманого з getUserMedia, іншим клієнтам, необхідно викликати метод addStream об'єкта RTCPeerConnection з потоком в якості параметра (рис. 2.12).

```
233 var Peer = new RTCPeerConnection(ICEServers, optional);
234
235 var constraints = { audio: true, video: true };
236 function success(stream) {
237     //Поток stream получен
238     Peer.addStream(stream);
239 }
240 function error() {
241     //Ошибка
242 }
243
244 navigator.webkitGetUserMedia(constraints, success, error);
245
```

Рисунок 2.12 - Відправка потоку в RTCPeerConnection

Після координації підключення і створення об'єкта RTCPeerConnection інший клієнт може отримати вхідний потік, коли спрацює подія onaddstream об'єкта RTCPeerConnection. Ця подія поверне об'єкт, що містить потік клієнта та інформацію про з'єднання (рис. 2.13).

```
248 Peer.onaddstream = function(event) {
249     var stream = event.stream;
250     var streamId = stream.id;
251
252     //Поток получен
253     //Отобразить в <video>:
254     video.src = webkitURL.createObjectURL(stream);
255
256     stream.onended = function() {
257         //Поток остановлен
258     };
259 };
```

Рисунок 2.13 - Отримання вхідного потоку

Таким чином можна організувати найпростіший сеанс потокової передачі даних, використовуючи основні методи Web API WebRTC.

2.3 Аналіз алгоритмів, що використовуються у WebRTC

2.3.1 Скасування акустичного відлуння за допомого алгоритму NLMS

LMS-алгоритм використовується в додатках, де апріорі можна оцінити гранично допустиме значення кроку збіжності. Така оцінка можлива, якщо відомі статистичні характеристики вхідного сигналах.

У додатках, де ці характеристики невідомі або змінюються з часом (наприклад, при обробці нестаціонарних сигналів), часто використовується нормалізований LMS-алгоритм (Normalized LMS, NLMS), що є однією з різновидів LMS-алгоритму.

NLMS-алгоритм [21] отриманий в результаті рішення задачі лінійнообмеженої оптимізації. NLMS-алгоритм розглядається як LMS- алгоритм зі змінним кроком збіжності, в якому цей крок визначається, виходячи з умови мінімізації з метою поліпшення швидкості збіжності алгоритму (формули 2.1-2.5).

$$\mathrm{X}_{N}(0) = 0_{N}, h_{N_x}(0) = 0_N \tag{2.1}$$

$$\mathrm{X}_{\mathrm{N_m}}(\mathrm{k})\big|2, \dots, N_m = X_{N_m}(k-1)\big|1, \dots, N_m - 1, X_{N_m}(k)|1 = x_m(k), m = 1{:}\,M \tag{2.2}$$

$$X_N(k) = \left[X_{N_1}^T(k), X_{N_2}^T(k), \dots, X_{N_m}^T(k), \dots, X_{N_M}^T(k)\right] \tag{2.3}$$

$$a_{N_x}(k) = d(k) - h_{N_x}^H(k-1)x_N(k) \tag{2.4}$$

$$h_{N_x}(k) = h_{N_x}(k-1) + \frac{\mu}{x_N^H(k)x_N(k)+\delta^2}x_N(k)\alpha_{N_x}^*(k) \tag{2.5}$$

Таблиця 2.5 – Позначення формули

$\mathbf{h}_{N,\chi}$	Вектор вагових коефіцієнтів багатоканального адаптивного фільтра.
$\mathbf{x}_{N}$	Вектор сигналів багатоканального адаптивного фільтра.
$\mathbf{x}_{N_m}$	Вектор сигналів в m -м каналі багатоканального адаптивного фільтра.
N	Число вагових коефіцієнтів адаптивного фільтра.
N_m	Число вагових коефіцієнтів в m -м каналі багатоканального адаптивного фільтра.
M	Число каналів багатоканального адаптивного фільтра.
k	Номер ітерації роботи адаптивного фільтра.
K	Число ітерацій роботи адаптивного фільтра.
d	Необхідний (desired) сигнал адаптивного фільтра.
x_m	Вхідний сигнал m -го каналу багатоканального адаптивного фільтра.

2.3.2 Робота jitter – буферу

Логіка роботи джиттер-буфера кінцевих абонентських пристроїв полягає в компенсації джиттера наскрізної затримки передачі пакетів шляхом їх буферизації протягом часу, достатнього для послідовного і своєчасного відтворення аудіо- та відеоінформації, як це було сформовано джерелом. При цьому допускається деякий відсоток втрат втрачених пакетів.

Моменти часу передачі пакетів від джерел суворо відомі і повинні бути синхронізовані в порядку проходження в приймальному абонентському пристрої. Стан процесу обробки пакетів в джиттер-буфері, визначається k-м номером пакету, що надійшов, включає дві фази - надходження пакета k в момент часу $t_b[k]$ і його зчитування в момент часу $t_{b'}[k]$. При цьому затримка в джиттер-буфері має довільний розподіл, і вона фактично визначається джиттером затримки і втратами пакетів в супутниковому каналі.

Завдання джиттер-буфера полягає у відновленні порядку проходження пакетів і їх міжпакетної тривалості, закладеної джерелом на передавальній стороні $\Delta t_{b'}[k] = \Delta t_a[k]$.

Інтервали між пакетами відновлюються на основі значень часових міток RTP-пакетів. Логіка роботи джиттер-буфера пов'язана з превентивним введенням часу утримання прийнятих пакетів $t_{b'}$[n], в момент закінчення якого значення обсягу джиттер-буфера одинакове (формула 2.6).

$$x_{qpl}(t) = \beta \max\left(\widehat{\mathrm{JD}}\,[k], 0\right) \widehat{\mathrm{R}}[k] \qquad (2.6)$$

Ініціалізується процес відтворення, де β - коефіцієнт надмірності, β> 1, оцінки джиттера затримки $JD[k]$ і бітової швидкості передачі $R[k]$ визначаються за результатами аналізу трафіку. Пакети, які прибули після закінчення моментів часу $t_{b'}$[n], вважаються втраченими.

Занадто короткий буфер буде приводити до частих втрат пакетів,що запізнились, а надто довгий - до неприйнятно великий додаткової затримки і погіршення інтерактивності людського спілкування. У зв'язку з цим параметри, що налаштовуються фактично визначатимуть компроміс між величиною затримки, і втратами пакетів.

2.3.3 Оцінка коефіцієнтів придушення шуму

Оцінка амплітудного спектра шуму [22] |N(k,m)| формується шляхом усереднення амплітуд попередніх відліків сигналу:

$$\left|\widehat{N}(k,m)\right| = \tilde{a}_d(k,m-1)\left|\widehat{N}(k,m-1)\right| + \left(1-\tilde{a}_d(k,m-1)\right)|\bar{X}(k,m)| \qquad (2.7)$$

де $\tilde{a}_d(k,m) = \alpha_d + (1-\alpha_d)\hat{p}(k,m)$, це параметр згладжування, що змінюється в часі, який залежить від умовної ймовірності присутності мовного сигналу $\hat{p}(k,m), \alpha_d$ - параметр згладжування, який визначає час усереднення, коли мова відсутня ($0 < \alpha_d < 1$).

$$G_{NR}(k,m) = max\left\{\sqrt{|\bar{X}(k,m)|^2 - v\left|\widehat{N}(k,m)\right|^2|\bar{X}(k,m)|^2}\,, 10^{-RL/20}\right\} \qquad (2.8)$$

де v - коефіцієнт вирахування (1< v < 6), RL - параметр, що визначає бажаний рівень залишкового шуму в дБ.

2.4 Удосконалення WebRTC

У розгортанні медіасерверів для WebRTC є дві складності: масштабування, тобто вихід за рамки використання одного сервера і оптимізація затримок для всіх учасників конференції. У той час як просте рішення «відправити всіх користувачів конференції X на сервер Y» легко масштабується горизонтально, він все ж далеко не оптимальний в плані затримок. Розподіляти конференцію по серверам, які не тільки близько розташовані до користувачів, а й взаємопов'язані - звучить як рішення для обох проблем.

Комунікації в реальному часі дуже чутливі до мережі: пропускна здатність, затримки і втрати пакетів. Зниження бітрейту веде до зниження якості відео, тривала мережева затримка веде до тривалої затримки у кінцевих користувачів. Втрата пакетів може зробити звук переривчастим і привести до фризів на відео (через пропуск кадрів).

Тому для конференції дуже важливо вибрати оптимальний маршрут між кінцевими пристроями / користувачами. Коли є тільки два користувача, то це просто - WebRTC використовує протокол ICE щоб встановити з'єднання між учасниками. Якщо можливо, то учасники з'єднуються безпосередньо, в іншому випадку використовується TURN-сервер. WebRTC вміє розпізнавати доменне ім'я, щоб отримувати адресу TURN-сервера, завдяки

чому можна легко вибирати локальний TURN на основі DNS. Проте, коли роутинг безлічі учасників відбувається через один центральний медіасервер, ситуація стає складною. Багато WebRTC-сервісів використовують Selective Forwarding Units (SFU), щоб більш ефективно передавати аудіо і відео між 3 і більше учасниками. Багато сервісів використовують простий підхід, який непогано працює в більшості випадків: вони вибирають сервер ближче до першого учаснику конференції. Однак бувають випадки, коли це рішення не оптимально. Метою роботи є підвищення якості та зниження затримки в конференціях WebRTC-сервісів.

WebRTC використовує RTP (зазвичай поверх UDP), щоб передавати медіа. Це означає, що транспорт ненадійний. Коли втрачається UDP-пакет, то можна ігнорувати втрату або запросити повторну відправку (ретрансмісію), використовуючи пакет RTCP NACK. Наприклад, додаток може проігнорувати втрату аудіопакетов і запросити ретрансмісію деяких (але не всіх) відеопакетів, в залежності від того, чи потрібні вони для декодування наступних кадрів чи ні (рис. 2.14).

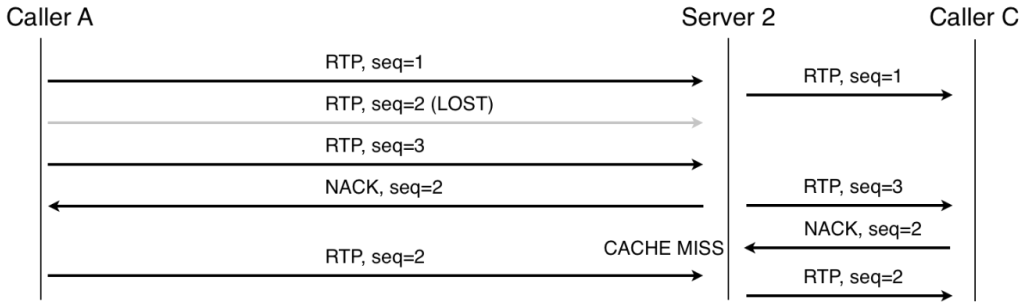


Рисунок 2.14 – Ретрансмісія RTP-пакету, 1 сервер

Коли є каскадування, ретрансмісія може бути обмежена локальним сервером, тобто виконуватися на кожній окремій ділянці. Наприклад, в маршруті A-S1-S2-C, якщо пакет втрачений між A і S1, то S1 це помітить і запросить ретрансмісію; аналогічно з втратою між S2 і C. І навіть якщо пакет втрачений між серверами, приймаюча сторона також може запросити ретрансмісію (рис. 2.15).

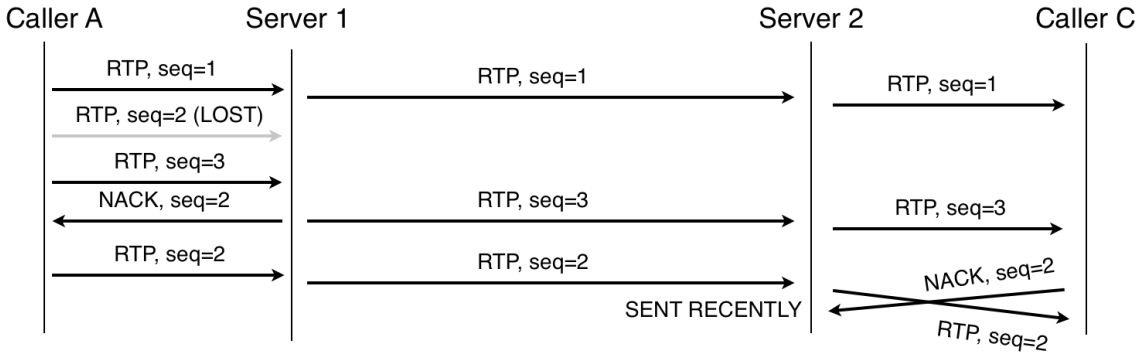


Рисунок 2.15 - Ретрансмісія RTP-пакету, два сервера

На клієнті використовується джиттер буфер, щоб затримати відтворення відео і встигнути отримати відкладені / ретрансмітні пакети. Розмір буфера динамічно змінюється в залежності від часу обміну між сторонами. Коли відбуваються hop-by-hop ретрансміссії, затримка зменшується, і як наслідок, буфер може бути менше - в результаті загальна затримка теж зменшується.

Щоб керувати взаємодією, було обернуто RTP-пакети в прості заголовки фіксованої довжини. У поточній реалізації, мости пов'язані повнозв'язною топологією (full mesh), проте можливі й інші топології. Замість обертання в заголовок можна використовувати розширення RTP-заголовків, яке зробить потоки між мостами на чистому (S) RTP. Коли міст - це частина конференції з множинними мостами, то у нього є додатковий канал, один канал на аудіо і один на відео. Цей канал відповідає за відправку / отримання медіа в / з інших мостів. Кожному мосту призначається вільний порт. Simulcast дозволяє кожному учаснику відправляти кілька медіапотоків з різними бітрейтами, в той час як міст допомагає визначити, які з них потрібні. Щоб це правильно працювало, ми передаємо всі simulcast-потоки між мостами. Однак це не оптимально з точки зору bridge-to-bridge трафіку, тому що деякі потоки рідко використовуються і лише навантажують смугу пропускання без будь-якої мети.

Для вирішення поставлених завдань було проаналізовано аналогічні сервіси та рішення, які вони використовують. При виборі основний акцент ставився на легкість масштабування та якість передачі медіа контенту. В результаті було обрано метод, який дозволяє легше масштабувати кількість учасників в конференції на WebRTC – додатку, зменшити затримку між ними та поліпшити якість. А саме було обрано обертання RTP-пакетів в заголовки фіксованої довжини та використання технології Simulcast, щоб відправляти кілька медіапотоків. Недоліком даного рішення можна назвати ресурсозатратність та збільшення кількості серверів для підтримки конференції.

2.5 Висновки до розділу 2

У даному розділі магістерської роботи проведено огляд, аналіз, дослідження та порівняння існуючих технологій для передачі інформації між двома вузлами мережі. А саме java, flash та WebRTC. Найбільш підходящою виявилась технологія передачі аудіо та відео контенту WebRTC. Було розглянуто основні алгоритми технології для роботи з відео та аудіо. Також було запропоновано метод каскадування для спрощення масштабування та поліпшення якості передачі контенту через WebRTC.

РОЗДІЛ 3
РОЗРОБКА І ТЕСТУВАННЯ ВЕБ – ДОДАТКУ ДЛЯ ОБМІНУ КОНТЕНТОМ

3.1 Побудова базового WebRTC-додатку

Для створення базового додатку на базі технології webrtc [23] потрібно створити з'єднання між двома користувача всередині однієї сторінки. Додаток буде виконувати послідовність дій:

1. створення 2-х RTCPeerConnection об'єктів;
2. створення SDP offer і передача його між ними;
3. пошук ICE кандидата для з'єднання;
4. установка WebRTC-з'єднання.

Простий шаблон html сторінки має бути з двома елементами video і кнопками для створення відеопотоку, виклику віддаленого абонента і завершення з'єднання (рис. 3.1).

```
<div id="container">
   <video id="localVideo" playsinline autoplay muted></video>
   <video id="remoteVideo" playsinline autoplay></video>

   <button id="startButton"> Start </button>
   <button id="callButton"> Call </button>
   <button id="stopButton"> Stop </button>
</div>
```

Рисунок 3.1 – HTML шаблон

Далі потрібно визначити змінні цих елементів, їх початковий стан, колбеки і блок функцій для дебага.

```
'use strict';

    const startButton = document.getElementById('startButton');
    const callButton = document.getElementById('callButton');
    const hangupButton = document.getElementById('stopButton');
    callButton.disabled = true;
    hangupButton.disabled = true;
    startButton.addEventListener('click', start);
    callButton.addEventListener('click', call);
    hangupButton.addEventListener('click', stop);
    let startTime;
    const localVideo = document.getElementById('localVideo');
    const remoteVideo = document.getElementById('remoteVideo');

    async function start() {
        console.log('Start')
    };

    async function call() {
        console.log('Calling')
    };

    async function stop() {
        console.log('Stop')
    };

    localVideo.addEventListener('loadedmetadata', function() {
      console.log(`Local video videoWidth: ${this.videoWidth}px,  videoHeight: ${this.videoHeight}px`);
    });

    remoteVideo.addEventListener('loadedmetadata', function() {
      console.log(`Remote video videoWidth: ${this.videoWidth}px,  videoHeight: ${this.videoHeight}px`);
    });
```

Рисунок 3.2 – Визначення початкових функцій та колбеки

Тепер потрібно визначити змінні для локального медіапотоку, об'єктів RTCPeerConnection, і параметрів offer запиту.

```
let localStream;
    let pc1;
    let pc2;
    const offerOptions = {
      offerToReceiveAudio: 1,
      offerToReceiveVideo: 1
    };

    function getName(pc) {
      return (pc === pc1) ? 'pc1' : 'pc2';
    }

    function getOtherPc(pc) {
      return (pc === pc1) ? pc2 : pc1;
    }
```

Рисунок 3.3 – Змінні для локального медіапотоку

Останні дві функції допоміжні, за поданим з'єднанням вони повертають протилежне з'єднання і його назву.

В функцію start потрібно додати відеопотік з локальної камери до елементу localVideo.

```
async function start() {
        console.log('Start')
        console.log('Requesting local stream');
        startButton.disabled = true;
      try {
          const stream = await navigator.mediaDevices.getUserMedia({audio: true, video: true});
          console.log('Received local stream');
          localVideo.srcObject = stream;
          localStream = stream;
          callButton.disabled = false;
      } catch (e) {
          alert(`getUserMedia() error: ${e.name}`);
      }
    };

```

Рисунок 3.4 – Додавання відео потоку

Після натискання кнопки Start в контейнері video відобразиться відео з камери.

Далі потрібно описати функцію call і те, що вона використовує (рис. 3.5).

addTrack (track, stream) додає нові медіадорожки в колекцію stream, які повинні бути передані клієнту на віддалений peer. Справа в тому, що по мережі передаються не потоки stream, а медіадорожки. Є можливість додати кілька доріжок з різних stream-об'єктів (у разі якщо є декілька камер і мікрофонів). Функція addTrack служить для їх угруповання і подальшої синхронізації. В данному прикладі використовується один об'єкт stream. Отримати поточні доріжки можна з об'єкта stream stream.getTracks (). Цей метод викликає подію negotiationneeded.

createOffer ([options]) формує об'єкт SDP offer для ініціалізації нового WebRTC-з'єднання. Цей об'єкт містить:

1. інформацію про всі медіадорожки, підключені в сесію WebRTC;
2. кодеки;
3. опції браузера;
4. ICE кандидати.

```
async function call() {
    callButton.disabled = true;
    hangupButton.disabled = false;
    console.log('Starting call');
    const videoTracks = localStream.getVideoTracks();
    const audioTracks = localStream.getAudioTracks();
    if (videoTracks.length > 0) {
      console.log(`Using video device: ${videoTracks[0].label}`);
    }
    if (audioTracks.length > 0) {
      console.log(`Using audio device: ${audioTracks[0].label}`);
    }

    const configuration = {};
    console.log('RTCPeerConnection configuration:', configuration);
    pc1 = new RTCPeerConnection(configuration);
    console.log('Created local peer connection object pc1');
    pc2 = new RTCPeerConnection(configuration);
    console.log('Created remote peer connection object pc2');

    pc1.addEventListener('icecandidate', e => onIceCandidate(pc1, e));
    pc2.addEventListener('icecandidate', e => onIceCandidate(pc2, e));

    pc2.addEventListener('track', gotRemoteStream);

    localStream.getTracks().forEach(track => pc1.addTrack(track, localStream));
    console.log(localStream.getTracks());

    try {
      console.log('pc1 createOffer start');
      const offer = await pc1.createOffer(offerOptions);
      await onCreateOfferSuccess(offer);
    } catch (e) {
       console.log(`${e}`);
    }
  }
```

Рисунок 3.5 – Функція дзвінка

Вся інформація призначена для передачі через сигнальний сервер на віддалений хост для конфігурації або поновлення віддаленого з'єднання. В параметрах options визначається які типи медіапотоку потрібно отримати параметрами offerToReceiveAudio і offerToReceiveVideo.

createAnswer ([options]) схожий на попередній метод createOffer, але викликається у відповідь на що прийшов SDP offer від віддаленого хоста.

setLocalDescription (SDP) і setRemoteDescription (SDP) - ці два методи встановлюють або оновлюють інформацію про локальну або видалену конфігурації медіапотоків. Зазвичай ці методи викликаються відразу після createOffer () і передачі SDP offer-а через сигнальний сервер з метою оновити інформацію.

addIceCandidate (). Коли додаток отримує ICE кандидата з віддаленого хоста його необхідно передати ICE агенту всередині об'єкта RTCPeerConnection. За це відповідає функція addIceCandidate (). Якщо приходить порожній рядок, це означає, що всі кандидати

були доставлені. Зазвичай таких кандидатів багато і кожен з них описує свій власний потенційний спосіб з'єднання.

У функції onIceCandidate додається переданий ICE кандидат в протилежне з'єднання, яке отримано функцією getOtherPc ().

```
async function onIceCandidate(pc, event) {
  try {
    await (getOtherPc(pc).addIceCandidate(event.candidate));
    onAddIceCandidateSuccess(pc);
  } catch (e) {
    onAddIceCandidateError(pc, e);
  }
  console.log(`${getName(pc)} ICE candidate:\n${event.candidate ? event.candidate.candidate : '(null)'}`);
}
```

Рисунок 3.6 – Додавання кандидату

Далі, викликаємо функціюцію onCreateOfferSuccess () першого з'єднання.

```
async function onCreateOfferSuccess(desc) {
  console.log(`Offer from pc1\n${desc.sdp}`);
  console.log('pc1 setLocalDescription start');
  try {
    await pc1.setLocalDescription(desc);
    onSetLocalSuccess(pc1);
  } catch (e) {
    console.log(`error setting description to pc1 ${error.toString()}`);
  }

  console.log('pc2 setRemoteDescription start');
  try {
    await pc2.setRemoteDescription(desc);
    onSetRemoteSuccess(pc2);
  } catch (e) {
    console.log(`error setting description to pc2 ${error.toString()}`);
  }

  console.log('pc2 createAnswer start');

  try {
    const answer = await pc2.createAnswer();
    await onCreateAnswerSuccess(answer);
  } catch (e) {
    onCreateSessionDescriptionError(e);
  }
}
```

Рисунок 3.7 – Формування виклику offer

У цій функції здійснюється 3 дії:

1. встановлюється setLocalDescription з SDP offer для pc1;
2. встановлюється setRemoteDescription з SDP offer для pc2;
3. формується SDP offer із з'єднання pc2.

Третій пункт викликається в останню чергу і встановлює SDP offer функціями pc2.setLocalDescription і pc1.setRemoteDescription SDP вже в зворотному напрямку.

```
async function onCreateAnswerSuccess(desc) {
      console.log(`Answer from pc2:\n${desc.sdp}`);
      console.log('pc2 setLocalDescription start');
      try {
        await pc2.setLocalDescription(desc);
        onSetLocalSuccess(pc2);
      } catch (e) {
        onSetSessionDescriptionError(e);
      }
      console.log('pc1 setRemoteDescription start');
      try {
        await pc1.setRemoteDescription(desc);
        onSetRemoteSuccess(pc1);
      } catch (e) {
        onSetSessionDescriptionError(e);
      }
    }
```

Рисунок 3.8 – Функція формування SDP offer

І також потрібно додати сервісні функції для дебагу.

```

  function onSetLocalSuccess(pc) {
    console.log(`${getName(pc)} setLocalDescription complete`);
  }

  function onSetRemoteSuccess(pc) {
    console.log(`${getName(pc)} setRemoteDescription complete`);
  }

  function onCreateSessionDescriptionError(error) {
    console.log(`Failed to create session description: ${error.toString()}`);
  }

  function onAddIceCandidateSuccess(pc) {
    console.log(`${getName(pc)} addIceCandidate success`);
  }

  function onAddIceCandidateError(pc, error) {
    console.log(`${getName(pc)} failed to add ICE Candidate: ${error.toString()}`);
  }

  localVideo.addEventListener('loadedmetadata', function() {
    console.log(`Local video videoWidth: ${this.videoWidth}px,  videoHeight: ${this.videoHeight}px`);
  });

  remoteVideo.addEventListener('loadedmetadata', function() {
    console.log(`Remote video videoWidth: ${this.videoWidth}px,  videoHeight: ${this.videoHeight}px`);
  });
```

Рисунок 3.9 – Функції для дебагу

Після натискання кнопки «START!» в браузері запитується дозволення на використання камери та мікрофону (рис. 3.10). Як дозвіл отримано – з'явиться відео з камери в вікні веб – додатку. Після натискання кнопки «CALL!» буде отримано зображення від іншого абоненту.

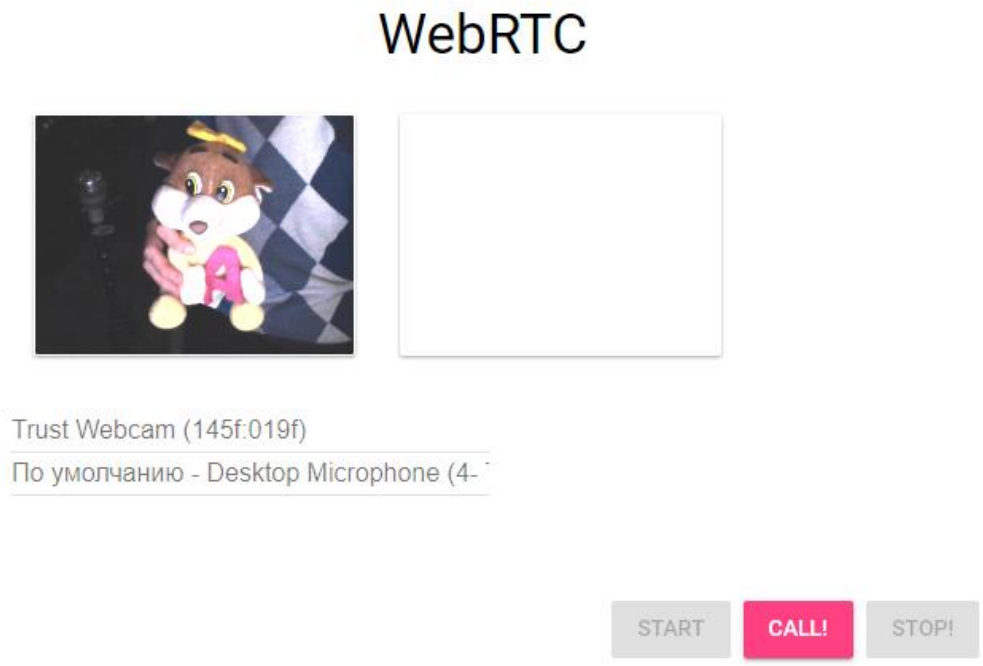


Рисунок 3.10 – Сторінка веб - додатку

3.2 Способи налагодження роботи додатку

Далі буде розглянуто основний інструментарій для вирішення найпопулярніших проблем зв'язаних з розробкою та впровадженням webRTC додатків.

3.2.1 Встановлення WebRTC підключення

Всі підключення WebRTC [24,25] відбуваються з допомогою сигнального протоколу. Сигнальний протокол - це сервер і протокол, за допомогою якого той, хто телефонує зможе поспілкуватися з тим, кому дзвонить, перш ніж встановити Peer-to-peer підключення.

WebRTC скористається сигнальним протоколом для передачі інформації про IP-адреси, можливості по захопленню і відтворення голосу і відео, топології мережі, переданих даних.

Зазвичай використовується протокол COMET (або SIP) і веб-сокети. WebRTC нічим

не обмежує розробників, так що можна використовувати будь-яку технологію для цього. Підключений до обох комп'ютерів сигнальний протокол дозволяє почати підключення вже по WebRTC.

3.2.2 Offer та answer

WebRTC підключення використовують «offer» і «answer»:

1. Ініціатор підключення створює і передає іншій стороні «offer»
2. Інша сторона отримує «offer», створює «answer» і передає його назад;
3. Ініціатор підключення отримує «answer».

На практиці обмін виглядає не так просто.

Перед передачею «offer» ініціатор підключення створює екземпляр RTCPeerConnection і отримує у нього текстовий пакет «SDP» (Session Description Protocol) за допомогою rtcPeerConnection.createOffer(); цей пакет визначає повноваження по прийому / передачі голосу і відео для браузера.

Вміст SDP-пакета встановлюється як «опис локальної боку підключення» за допомогою rtcPeerConnection.setLocalDescription().

Пакет передається іншій стороні, де його вміст встановлюється як «опис іншого боку підключення» за допомогою rtcPeerConnection.setRemoteDescription().

На іншій стороні підключення створюється власний SDP-пакет за допомогою rtcPeerConnection.createAnswer (), його вміст встановлюється в якості «опису локальної боку підключення».

Пакет передається ініціатору підключення, який встановлює його вміст як «опис іншого боку підключення».

І тільки після всіх дій абоненти знають можливості один одного по прийому й відправленню голосу / відео.

3.2.3 ICE-кандидати

Одних можливостей по роботі з медіаданими мало. Адже сторони ще нічого не сказали про стан мережі.

Дізнатися, які видеокодеки підтримує браузер і чи є на ноутбуці камера, можна майже миттєво. Для того, щоб дізнатися свій зовнішній IP-адреса і логіку роботи NAT потрібен час, і обмін інформацією про стан мережі відбувається в міру отримання цієї

інформації.

Завдяки технології Trickle ICE підключення між двома WebRTC-пристроями може встановитися в будь-який момент - як тільки буде знайдений відповідний «кандидат».

Розробник повинен підписатися на подію onicecandidate і передавати іншій стороні одержувані SDP-пакети, де їх потрібно передавати WebRTC за допомогою методу addIceCandidate.

3.2.4 STUN-сервер

Коли учасники підключення хочуть описати як до них підключатися, їм потрібен їх публічний IP-адрес. І швидше за все це не буде IP-адреса комп'ютера, так як користувачам пристроїв рідко виділяють публічні адреси. Для цього використовується технологія NAT. Щоб все-таки дізнатися свою публічну адресу, браузер робить запит до STUN сервера. Проходячи через NAT, мережевий пакет змінює свою зворотну адресу на публічну. Отримавши пакет із запитом, STUN-сервер [27] копіює зворотну адресу пакета в його payload і відправляє пакет назад. Проходячи через NAT в зворотну сторону, пакет втрачає публічну IP-адресу, але копія цієї адреси залишається в payload, де її може прочитати WebRTC.

3.2.5 TURN-сервер

TURN-сервер [28] використовує розширення STUN-протоколу. Ті ж пакети, заголовки. Сервер є проксі: обидва клієнта підключаються до нього через UDP-порт allocation і передають через сервер свої дані.

TURN-сервери влаштовані таким чином, що ініціатор підключення має більше можливостей, ніж інша сторона. Це призводить до цікавого ефекту, коли дзвінок через TURN сервер буде успішним або неуспішним в залежності від того, хто кому дзвонить.

3.2.6 Локальне налогодження webRTC

Основна робота WebRTC відбувається на стороні браузера. STUN і TURN-сервери неймовірно прості, так що більшість проблем трапляються у JavaScript-коді, який виконується в двох браузерах.

Перше що треба перевірити - це сигналювання. Саме код передає між браузерами

конфігурацію аудіо з відео (offer, answer) і інформацію про мережні параметри (ice кандидати). Потрібно перевірити, які пакети були відправлені, які отримані і передані WebRTC:

Перевірити отримання offer,перевірка отримання answer.

WebRTC на обох кінцях підключення повинен передати пакети з ICE-кандидатами.

Пакети Offer, Answer і ICE-кандидатів створюються WebRTC в текстовому форматі SDP. Найважливіше поле в SDP пакетах ICE кандидатів це typ. Для WebRTC поле може мати одне з трьох значень:

1. typ host;
2. typ srflx;
3. typ relay.

Тип host задає ICE-кандидат на підключення по локальній мережі. WebRTC перебирає кілька кандидатів в надії встановити підключення, заздалегідь невідомо, який вийде. Таке підключення не вимагає ні STUN, ні TURN-сервера, так як в локальній мережі пристрою часто можуть встановлювати мережеві підключення безпосередньо. При налагодженні з локальної мережі досить перевірити і налагодити передачу host-пакетів і переконатися, що пристрої можуть надсилати один одному UDP-пакети. Хоча бувають винятки, на практиці бувають конфігурації мережі, при яких браузеру потрібний TURN-сервер, щоб підключитися до себе.

Комбінація букв «srflx» розшифровується як «Server Reflexive» і маркує кандидатів на підключення з використанням зовнішньої IP-адреси, де для підключення досить STUN-сервера, при цьому використовується технологія NAT penetration, яка успішна приблизно в 80% випадків .

«Relay» маркує підключення через TURN-сервер, яке майже завжди успішно. Важливо пам'ятати, що WebRTC не повинно створити рівно три різних пакети з полем «typ»; як саме вибираються кандидати - залежить від імплементації WebRTC в конкретної версії браузера.

3.2.7 WebRTC Internals

Google забезпечила свій браузер Chrome спеціальним розділом, який показує нутрощі WebRTC під час установки підключення і графіки в разі успішного підключення. Для того, щоб отримати доступ до цього інструменту, потрібно ввести в браузері посилання chrome: // webrtc-internals .

Якщо вже використовується WebRTC, то всі розділи будуть заповнені даними. Вкладка відображає всі виклики до об'єкта RTCPeerConnection і дозволяє побачити в реальному часі як встановлюється підключення.

Зверху сторінки відображається ICE-рядок, який був використаний при ініціалізації підключення. Якщо при формуванні була допущена помилка, це відразу ж буде видно (рис. 3.11)

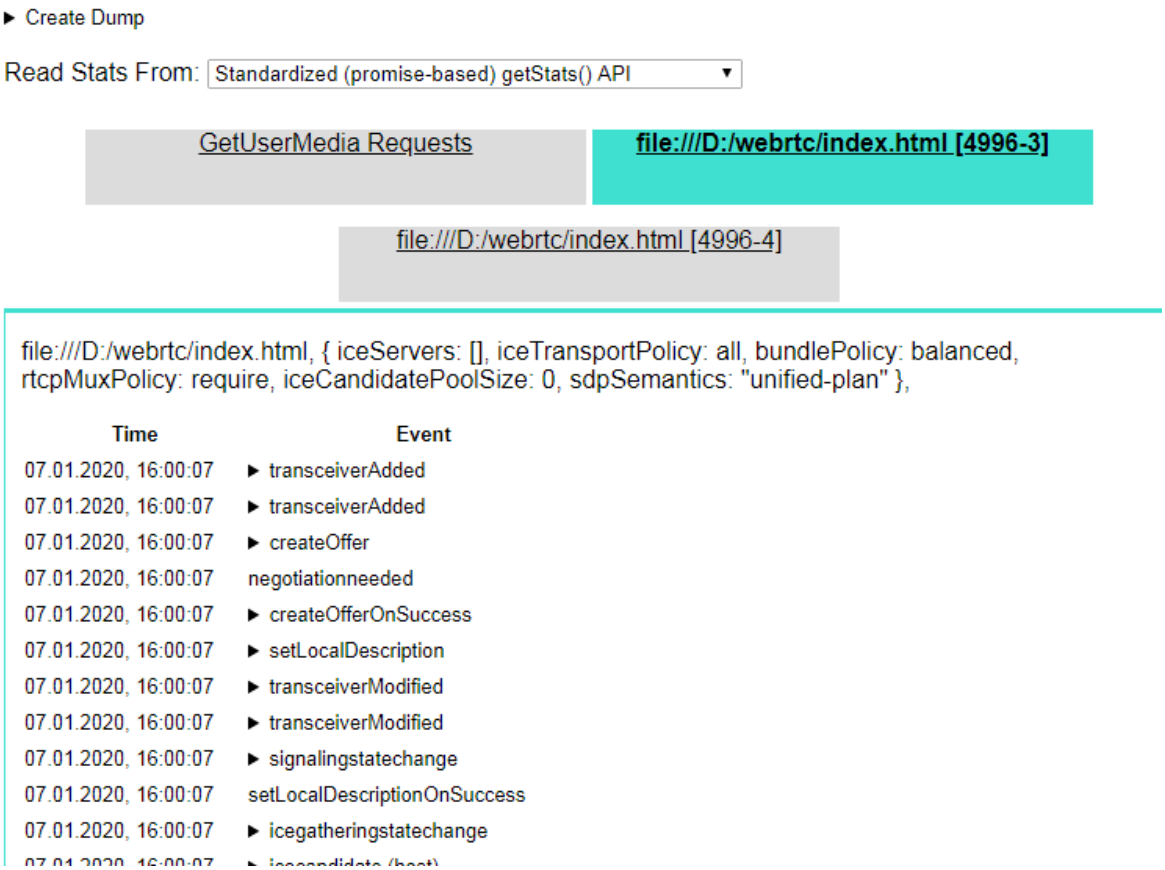


Рисунок 3.11 - Події RTCPeerConnection

Наступний розділ internals показує виклики методів RTCPeerConnection (рис. 3.12) і отримані від об'єкта події в хронологічному порядку. У разі успішного підключення останнім у списку буде подія iceconnectionstatechange з значенням complete. Наступний розділ актуальний, коли підключення успішно встановлено. Він містить статистику переданих даних і мережеві затримки. Два найбільш цікавих параметра: ssrc і bweforvideo.

ssrc, «Stream Source», маркує кожен звуковий ряд і відеотрек. Відображає статистику переданих даних і такі параметри як round trip time;

bweforvideo, BandWidth Estimate, відображає ширину використовуваного мережевого каналу.

Рисунок 3.12 – Розділ Stats

3.2.8 Зовнішнє налагодження

Крім самого об'єкта RTCPeerConnection і internals, що відображає браузер, можна використовувати інструменти аналізу мережевих пакетів, такі як Wireshark. Ці інструменти вміють відображати пакети використовуваних WebRTC-протоколів. Наприклад, Wireshark покаже вміст STUN-пакетів в головному вікні, і їх можна відфільтрувати, набравши в поле фільтра ключове слово «stun»:

Якщо всі пакети з типом відповіді Binding, то це означає тільки підтримку STUN, і WebRTC зможе запропонувати тільки srflx-кандидатів. Якщо ж у відповідях є специфічні для TURN пакети Allocation і CreatePermission, то у WebRTC буде можливість спробувати підключитися через проксі-сервер. Аналізатор пакетів позначає успішні і неуспішні Allocation. Якщо немає жодного успішного, то швидше за все передані невірні параметри доступу до TURN-серверів.

Якщо в программі є пакет CreatePermission Success Response, то можна припустити, що зі зміною STUN і TURN все добре. А якщо присутній ChannelBind пакет, то значить вдалося встановити підключення до TURN-серверу на високій швидкості.

3.3 Схема реалізації SFU каскадування у веб - додатку

Для реалізації каскадування потрібно принаймні декілька серверів. Щоб реалізувати множинні SFU потрібно використовувати хмарні технології , наприклад, Amazon AWS (рис. 3.13).

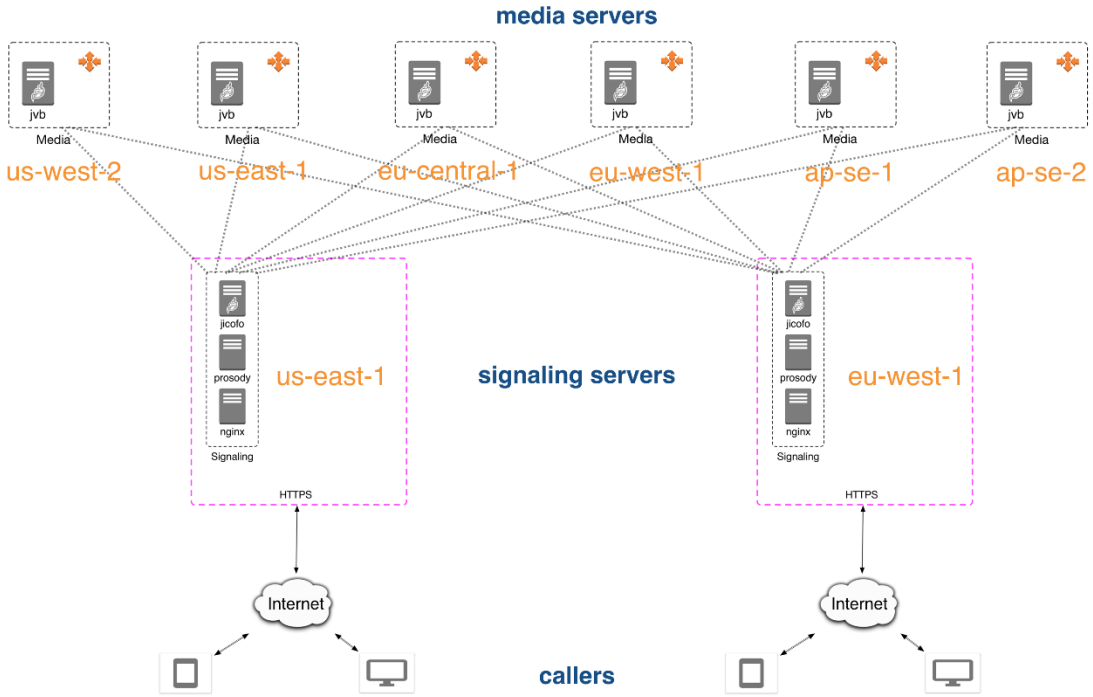


Рисунок 3.13 - Приклад організації серверів на AWS з можливістю каскаду між різними дата-центрами.

3.4 Тестування роботи веб – додатку та серверу

Тестування додатку та серверу відбувалося непосредньо в умовах реального використання. Спочатку було приєднано 3 співрозмовники з різних браузерів,а саме: два Chrome та один Yandex браузери. З’єднання всіх пристроїв відбулося успішно (рис. 3.14).

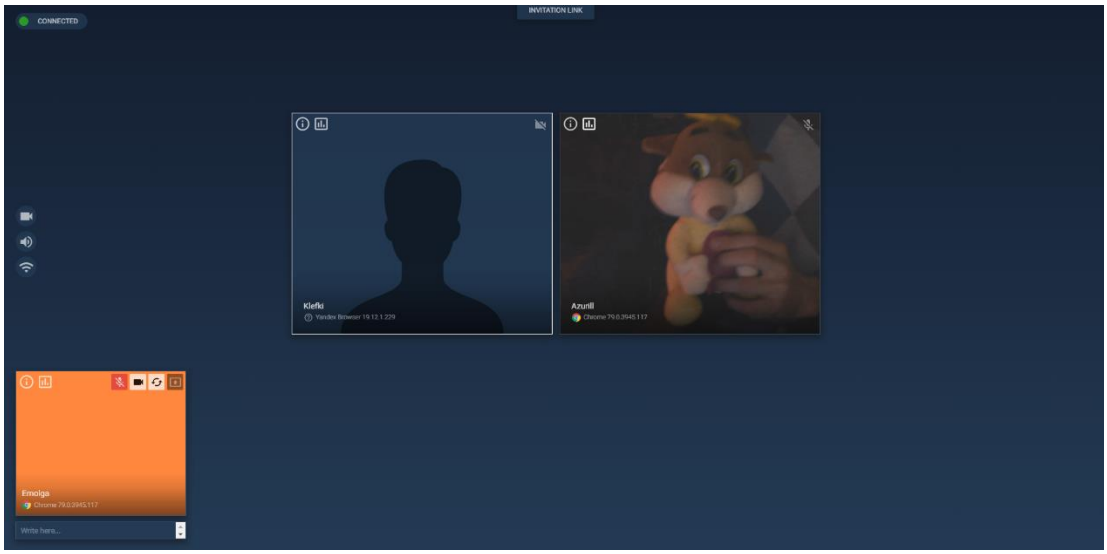


Рисунок 3.14 – Вікно конференції (3 учасника)

При натисканні на кнопку інформації про пристрій висвічується інформація про використовувані кодеки та інша інформація про підключення (рис. 3.15).

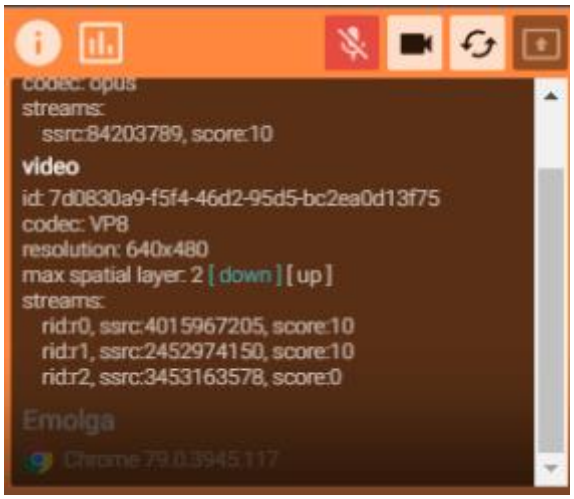


Рисунок 3.15 – Інформація про пристрій та кодеки

3.5 Статистика підключення абонентів конференції

В додатку присутня статистика по підключенню, відправленим пакетам, бітрейту та іншій інформації (рис. 3.16, 3.17).

Рисунок 3.16 – Статистика підключення (send)

Рисунок 3.17 – Статистика підключення (recv)

Також доступна інформація про зовнішній IP учасників конференції та внутрішній IP конференції (рис. 3.18).

Рисунок 3.18 – ICE Candicate

В статистиці додатку також є інформація що до затримки jitter – буферу (рис. 3.19).

Рисунок 3.19 – Статистиика jitter - буферу

3.6 Тестування кросплатформності веб - додатку

Під час тествування було створено різні конфігурації конференції співрозмовників. В конференції приймали участь різні браузери та пристрої, а саме:

- chrome 79.0.3945.117 з операційною системою windows 10;
- chrome 79.0.3945.117 з операційною системою windows 7;
- firefox 69.0 з операційною системою windows 10;
- chrome 74.0.3729.136 з операційною системою android 9;
- yandex browser 19.12.1.229 з операційною системою windows 10;
- opera 65.0.3467.78 з операційною системою windows 10;
- safari 7.0.3 з операційною системою windows 10;
- safari 13.0.4 з операційною системою IOS 13;
- firefox 67.0 з операційною системою ubuntu;

Приклад однієї із конференцій на рисунку 3.20.

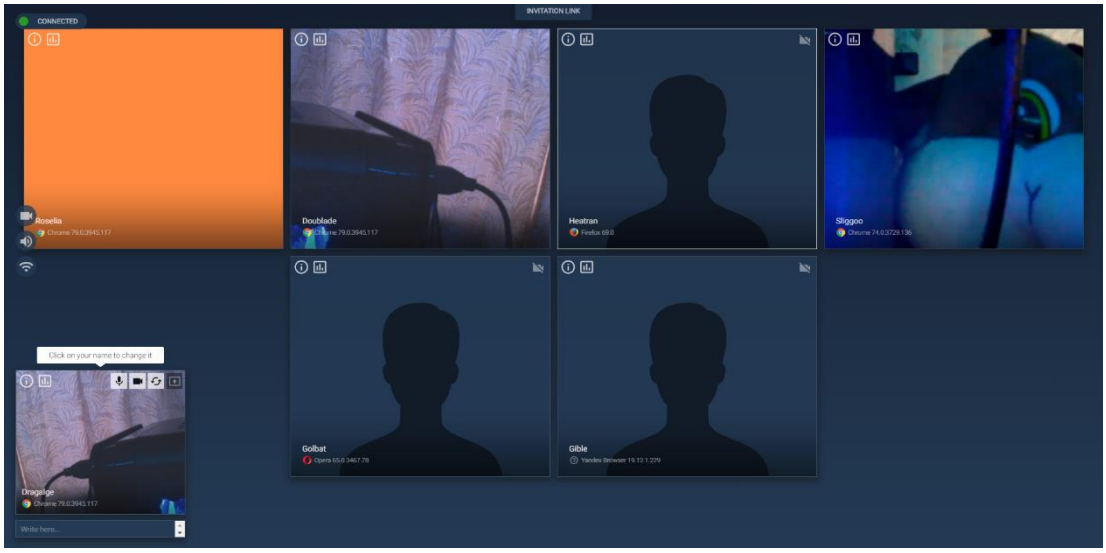


Рисунок 3.20 – Конференція з різними пристроями

Таким чином вдалося з’єднати всі пристрої на базі популярних браузерів та операційних систем. Використовуючи інтернет ресурс caniuse.com було отримано таблицю підтримуваних браузерів (рис. 3.21).

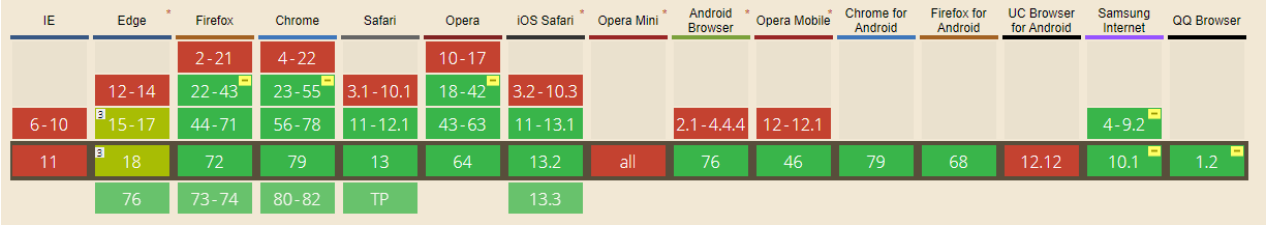


Рисунок 3.21 – Таблиця підтримуваних браузерів

Як видно з таблиці, браузер internet explorer не підтримує WebRTC, тому додати цей браузер до конференції не вдалося. Також мобільний браузер opera mobile та UC browser не підтримують роботу з WebRTC. Виходячи із цього можна сказати, що операційна система ніяк не впливає на роботу браузера та безпосередньо на функціювання самого WebRTC.

3.7 Висновки до розділу 3

У третьому розділі магістерської роботи було побудовано базовий WebRTC додаток, розглянуто роботу його компонентів, таких як ICE-кандидати, STUN та TURN сервери. та методи тестування, а саме: локальне та зовнішнє налагодження. До локального можна віднести розбір SDP протоколу та панель розробника internals в браузері Google Chrome. До зовнішнього тестування можно віднести різні аналізатори мережевих пакетів, наприклад, Wireshark, який дозволяє відстежити стан пакетів, що передаються.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

У цьому розділі аналізуються потенційно небезпечні та шкідливі виробничі фактори. На основі цього аналізу розроблені технічні, санітарно-гігієнічні та організаційні заходи, спрямовані на усунення причин виробничого травматизму, професійних захворювань, підвищення продуктивності праці та зменшення впливу на навколишнє середовище. Оскільки реалізація призначення випускного проекту здійснюється на ПК, то для комп'ютерних технологій проводиться аналіз потенційно небезпечних та шкідливих виробничих факторів. При роботі з комп'ютерною технікою змінюються фізико-хімічні фактори навколишнього середовища: відбувається статична електрика, електромагнітне випромінювання, зміна температури і вологості, рівень кисню і озону в повітрі. Також, невиконання вимог безпеки призводить до того, що при роботі з комп'ютером співробітник може відчувати дискомфорт: з'являються головні болі і розріз в очах, з'являються втому і дратівливість. Деяких людей турбують сон, апетит, зір, очі, рани, шия, поперековий та інші. При ненормальній роботі можливе нервове виснаження. Надання цих умов покладається на власника або уповноважений орган (далі - роботодавець). Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, обладнання та інших засобів виробництва, стан колективного та індивідуального захисту працівників, а також санітарні умови повинні відповідати вимогам нормативних актів про працю охорони, які чітко регулюються законом України.

4.1 Аналіз стану умов праці

Обчислювальна техніка при функціонуванні має наступні експлуатаційні характеристики:

- робоче живлення 220 В;
- частота живильної мережі 50 Гц;
- споживана потужність в межах 300 Вт.

При роботі на персональних ПЕОМ користувач наражається на небезпеку ураження електричним струмом. Приміщення для обчислювальної техніки за ступенем небезпеки ураження людини електричним струмом відноситься до приміщень без підвищеної небезпеки [29]. Тяжкість роботи персоналу, що обслуговує і працює на ПЕОМ, відноситься

до категорії 1а - легкі фізичні навантаження. При обслуговуванні обчислювальної техніки мають місце фізичні і психофізіологічні небезпечні та шкідливі виробничі фактори:

- підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини;
- підвищена або знижена температура повітря робочої зони;
- підвищена або знижена рухливість повітря;
- підвищена або знижена вологість;
- підвищений рівень електромагнітних полів у робочій зоні;
- відсутність або нестача природного світла;
- підвищена пульсація світлового потоку;
- розумове перенапруження;
- монотонність праці;
- емоційні навантаження;
- підвищений рівень шуму;
- недостатнє освітлення робочого місця;
- підвищена статична електроенергія.

4.2 Вимоги до приміщення

Для зручності спільної роботи з іншими працівниками (обговорення ідей, з'ясування проблем і т.д.) в кімнаті є диван і журнальний стіл. Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення та кондиціонування.

Згідно до санітарних норм мікроклімату виробничих приміщень розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм – неменше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Щоб захистити людей від ураження електричним струмом, коли вони вмирають до металевих нестійких деталей, які можуть піддаватися впливу напруги в результаті ізоляції, передбачаються наступні заходи:

- захисне заземлення або занулення металевих частин електроустановок, які доступні для дотику людини й не мають інших видів захисту, що забезпечують електробезпеку:
- захисне відключення;
- електричний поділ мереж;
- використання малої напруги;

- ізоляція струмоведучих частин;

- огорожу електроустановок;

- шина заземлення виконується провідником з опором не більше 4-х Ом.

Завданнями наземного забезпечення безпеки є усунення небезпеки ураження електричним струмом шляхом допуску живлення до корпусу та інших поточних металевих частин електроустановки.

Розрахунок заземлюючого контуру виконується виходячи з умови:

$$R_з = \frac{R_з * R_з}{R_П * n * ŋ_з * R_з * ŋ_П} \leq 4 \text{ Ом}, \quad (4.1)$$

де $R_З$ - опір заземлювача (стержня, труби, куточка і т.д.), Ом;

$R_П$ - Опір лінії, що з'єднує заземлювачі, Ом;

n - кількість заземлювачів;

$ŋ_з$ і $ŋ_П$ - Коефіцієнти екранування відповідно заземлювача і з'єднує смуги ($ŋ_З$= 0,2 ÷ 0,9; $ŋ_П$ = =0,1÷0,7).

Опір заземлювача розраховується за формулою 1.2

$$R_з = \frac{\rho}{2*\pi*l}\left(\ln\frac{2*l}{d} + \frac{1}{2} * \ln\frac{4*t+1}{4*t-1}\right), \quad (4.2)$$

де ρ - питомий опір ґрунту (взяти з довідкової літератури);

l - довжина заземлювача (для труб 2-3 м, для стрижнів до 10 м), м;

d - діаметр заземлювача (для стрижнів 0,01 - 0,03 м, для труб 0,03 - 0,05 М);

t - відстань від середини забитого в ґрунт заземлювача до рівня землі (необхідно враховувати, що відстань від верхнього кінця заземлювача до поверхні землі має бути не менше 0,5), м.

Розрахуємо опір заземлювача:

$$R_з = \frac{60}{2*\pi*3}\left(\ln\frac{2*3}{0.03} + \frac{1}{2} * \ln\frac{4*1+3}{4*1-3}\right) = 19.96,$$

Опір лінії, що з'єднує заземлювачі розраховується за формулою 4.4

$$R_{П}=\frac{\rho}{2*\pi*l}*\ln\frac{2*L^{2}}{b*t}, \quad (4.3)$$

де L - довжина лінії, що з'єднує заземлювачі (при контурному заземленні вона приблизно дорівнює периметру виробничої будівлі), м;

b - ширина смуги (0,03 - при прокладанні всередині будівлі і 0,05 - при прокладанні поза будівлею), м;

t - глибина заземлення від рівня землі (0,5 м.).

Розрахуємо опір лінії, що з'єднує заземлювачі

$$R_{П}=\frac{60}{2*\pi*3}*\ln\frac{2*50^{2}}{0.03*5}=14.37,$$

Необхідна кількість заземлювачів, розраховується за формулою 1.6

$$n=\frac{2*R_{з}}{4*\eta_{з}}, \quad (4.4)$$

де 4 - допустимий загальний опір;

2 - коефіцієнт сезонності.

Розрахуємо необхідну кількість заземлювачів:

$$n=\frac{2*19.9}{4*0.5}=19.9\approx 20,$$

Округлимо результат в більшу сторону і отримуємо необхідну кількість заземлювачів - 20. Маючи всі необхідні дані розрахуємо опір заземлюючого контуру.

$$R_{з}=\frac{19.96*14.37}{14.37*20*0.5+19.96*0.4}=1.89\leq 4 \text{ Ом},$$

Опір заземлюючого контуру 1,89 Ом, що відповідає умові Rз < 4 Ом.

4.3 Вимоги до організації робочого місця

У кабінеті в зоні дослідження є достатньо місця для ніг. Крісло, яке використовується як робоче місце - це підйом-поворот, підлокітники і можливість регулювати висоту і кут задньої частини спини, він також м'який і виготовлений з екологічної шкіри, що дозволяє працювати в комфорті.

Екран монітора знаходиться на відстані 0,8 м, клавіатура має можливість регулювати кут нахилу 5-15°. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам.

Кімната офісу розташована на четвертому поверсі чотириповерхового будинку і має об'єм 87,5 м3, площею 25 м2. Цей офіс обладнаний шістьма робочими місцями, включаючи чотири повні комп'ютери. Температура в приміщенні протягом року коливається у межах 18–24°C, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум на робочому місці знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

Вікна забезпечені природним освітленням з природною інтенсивністю світла не менше 1,5%, а загальне штучне освітлення, яке здійснюється за допомогою восьми люмінесцентних ламп, забезпечує рівень освітленості не менше 200 лк.

У офісі є електрична мережа 220 В, що створює небезпеку ураження електричним струмом. ПК і периферійні пристрої можуть бути джерелами електромагнітного випромінювання, аерозолями і шкідливими речовинами (частинки тонера, оксидів азоту і озону).

За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет має бути оснащений переносним вуглекислотним вогнегасником ВВК-5.

Наявна аптечка для надання долікарської допомоги, а також у кабінеті роблять вологе прибирання та щоденно провітрюють приміщення.

4.4 Рекомендації із пожежної профілактики

Пожежа на робочому місці є небезпекою, оскільки вона пов'язана як з матеріальними втратами, так і з відмовою комп'ютерного обладнання.

Пожежа може виникнути, коли джерело займання в горючих умовах. Горючі матеріали в приміщеннях, де розташовані обчислювальні засоби, - це будівельні матеріали, віконні рами, двері, підлоги, меблі, ізоляція силових і сигнальних кабелів, радіодеталі,

конструктивні елементи пластичних матеріалів, рідини для чищення елементів і компоненти ПК забруднюючих речовин.

поліамід - матеріал корпусу мікросхеми, горюча речовина, температура самозаймання 420°С;

- полівінілхлорид - ізоляційний матеріал, горюча речовина, температура займання 335°С, температура самозаймання 530°С, теплота згоряння 18000-20700 кДж/кг;

- склотекстоліт ДЦ - матеріал друкованих плат, важко-горючий матеріал, показник горючесті 1.74, не схильний до температурного самозаймання; пластик кабельний № 489 - матеріал ізоляції кабелю, горючий матеріал, показник горючості більше 2.1;

- деревина - будівельний і оздоблювальний матеріал, матеріал з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, теплота згоряння 18731-20853 кДж/кг, температура займання 399°С, схильна до самозаймання [30].

Приміщення належить до категорії "В", як пожежонебезпечна. Приміщення відноситься до класу П-IIа. Пожежа може виникнути в результаті утворення джерела запалювання (іскри й дуги короткого замикання, порушення ізоляції, що приводить до короткого замикання, перегріву радіодеталей внаслідок тривалого перевантаження) та внесення його до горючої середи. При повному згорянні органічних сполук утворюється CO_2, SO_2, H_2O, N_2, а при згорянні неорганічних сполук - оксиди. Залежно від температури плавлення продукції реакції диму можуть або знаходитися у воді розплаву (Al_2O_3, TiO_2), або підійматися в повітря у вигляді диму (P_2O_5, Na_2O, MgO). Розплавлені тверді частинки створюють світимість полум'я. Склад продуктів неповного згоряння горючих речовин складний і різноманітний. Це можуть бути горючі речовини - H_2, CO, CH_4 та ін.; атомарний водень і кисень; різні радикали - OH, CH і інші. Продуктами неповного згорання можуть бути також оксиди азоту, спирти альдегіди, кетони й високотоксичні з'єднання, наприклад, синильна кислота.

Для захисту персоналу від можливого отруєння проектом передбачається застосування протигаза з коробкою марки «В» (жовта) [31].

Пожежна безпека при застосуванні ЕОМ забезпечується:

- системою запобігання пожеж;

- системою протипожежного захисту;

- організаційно-технічними заходами.

До системи запобігання пожежі відносяться:

- запобігання утворенню горючого середовища;

- електроживлення ПЕОМ має автоматичне блокування відключення електроенергії на випадок зупинки системи охолодження й кондиціонування;

- система вентиляції обчислювальних центрів обладнуються блокувальними пристроями, що забезпечують її відключення в разі пожежі;

- система обладнується вогнеперепинюваченими клапанами;

- застосування обладнання задовольняє вимогам;

- забезпечення пожежної безпеки обладнання;

- після закінчення роботи, перед закриттям приміщення, все електроустановки відключаються і ПК від мережі електроживлення.

Зменшити горюче навантаження не є можливим, тому проектом передбачається застосування наступних способів і їх комбінацій для запобігання утворення (внесення) джерел запалювання:

- застосоване обладнання задовольняє вимогам електростатичної безпеки;

- застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалювання;

- виключення можливості появи іскрового заряду статичної електрики в займистому середовищі з енергією, що дорівнює і вище мінімальної енергії запалювання;

- підтримання температури поверхневого нагріву машин обладнання механізмів речовин і матеріалів, які можуть контактувати з горючим середовищем нижче максимально допустимого, що становить 80% від найнижчої температури самозаймання палива.

Для зниження пожежної небезпеки проектом передбачається використовувати систему автоматичної пожежної сигналізації з одним димовим датчиком-оповіщувачем типу ВДМ-1М, який розрахований для контролю площі до 100 м^2 при висоті стелі до чотирьох метрів, а також первинні засоби пожежогасіння. В якості первинних засобів пожежогасіння передбачається використовувати:

- ручний вуглекислотний вогнегасник ОУ-5 - 1 шт .;

- повітряно-пінний вогнегасник ОВП-5 - 1 шт .;

- азбестове полотно 1.5 × 2 м.

В якості організаційно-технічних заходів рекомендується проводити навчання робочого персоналу пожежної безпеки.

4.5 Мікроклімат

У виробничому приміщенні на тілі людини і його працездатності впливають мікрокліматичні фактори. Мікроклімат виробничих приміщень визначається поєднанням температури, вологості і швидкості повітря, а також температури навколишніх поверхонь. В даний час основним нормативним документом, що стосується нормалізації мікроклімату, є [32].

Для робіт категорії 1а для робочої зони виробничих приміщень забезпечуються наступні метеорологічні умови:

- в холодний і перехідний період року температура повітря 22 ÷ 24°С;

- відносна вологість повітря 40 ÷ 60%, швидкість руху повітря не більше 0.1 м/с;

- у теплий період року температура повітря 23÷25°С, відносна вологість повітря 40-60%, швидкість руху повітря не більше 0.1 м/с.

Рівень шуму не перевищує санітарних норм. Тому застосування захисту від шуму в роботі не передбачається.

Інша проблема полягає в тому, що спектр випромінювання комп'ютерного монітора включає в себе рентгенівські, ультрафіолетові і інфрачервоні області, а також широкий спектр хвиль інших частот. Небезпека рентгенівських променів незначна, оскільки цей вид випромінювання поглинається речовиною екрану. Проте велика увага повинна приділятися біологічному впливу низькочастотних електромагнітних полів.

З метою зменшення втоми персоналу в приміщеннях, де розташовані обчислювальні засоби, передбачається використовувати кольорові комбінації та покриття, які не дають відблисків.

Очікується, що конструкція буде використовувати освітлення підключення. У світлий час дня кімната буде покрита через віконні прорізи, в інший час буде використано штучне освітлення. Штучне освітлення виробляється лампами розжарювання або газорозрядних ламп. Передбачається, що штучне освітлення в робочій зоні здійснюватиметься з використанням люмінесцентних джерел світла в звичайних світильниках, оскільки люмінесцентні лампи мають високу світлову ефективність (до 75 л / Вт і більше), тривалий термін служби (до 10 000 годин) , спектральний склад випромінюваного світла, близький до сонячного При експлуатації комп'ютерів візуальна робота IV виконана в категорії точності (середньої точності). У цьому випадку нормалізована освітленість на робочому місці становить 200 люкс. Джерелом природного світла (освітлення) є сонце. У приміщенні, де розташовані комп'ютери, існує природне

бічне освітлення, рівень якого відповідає ДБН В.2.5-28-2018 " Природнє і штучне освітлення" [33].

Регулярно проводиться контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для данного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок штучного освітлення заснований на коефіцієнтах використання світлового потоку, який визначає потік, необхідний для створення заданого освітлення з загальним рівномірним освітленням.

Розрахунок кількості світильників здійснюється за формулою:

$$N=\frac{E\cdot l\cdot m\cdot Z\cdot K}{U\cdot M\cdot F} \quad (4.5)$$

де N - число світильників;

E - нормоване освітлення;

S - площа підлоги, $м^2$, S=25 $м^2$;

Z - поправний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,2 для люмінесцентних ламп) приймаємо рівним 1,2;

K - коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U - коефіцієнт використання, що залежить від типу світильника, показника індексу приміщення і т. п. - 0,575;

M - число люмінесцентних ламп у світильнику - 2;

F - світловий потік – 5400лм (для ЛБ - 80).

Згідно вимог ДБН В.2.5-28-2018 [33], освітлення робочого місця оператора обчислювальної техніки повинно бути не менше 200 лк.

$$N=\frac{200\cdot 5\cdot 5\cdot 1{,}2\cdot 1{,}5}{5400\cdot 0{,}55\cdot 1}\approx 3$$

Обираємо кількість світильників, що дорівнює 3.

4.6 Охорона навколишнього природного середовища

Основним екологічним аспектом у процесі діяльності в цих спеціальностях є процеси впливу на атмосферне повітря і процеси поводження з відходами, які формуються, збираються, розміщуються, передаються у розпорядження, утилізацію та ін.

Вплив на атмосферне повітря при нормальних умовах роботи не працює, оскільки не має в приміщенні сканерів, принтерів та інших джерел викидів забруднюючих речовин в повітря в робочій зоні.

В процесі діяльності комп'ютера виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

1. відпрацьовані люмінесцентні лампи - I клас небезпеки;
2. батарейки та акумулятори (малі) -III клас небезпеки;
3. змінні носії інформації - IV клас небезпеки;
4. відходи друкуючих пристроїв - IV клас небезпеки;
5. відпрацьований ізолюючий матеріал, дроти та кабелі - IV клас небезпеки;
6. макулатура - IV клас небезпеки;
7. побутові відходи - IV клас небезпеки.

Відходи, оскільки їх накопичення збираються в контейнері, що відповідає класу небезпеки, з дотриманням правил безпеки, а потім доставляються до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення. Розміщені для зберігання відходи або об'єкти повинні використовуватися лише для оголошених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- Відходи I класу небезпеки зберігаються в герметичній тарі (сталеві бочки, контейнери). У міру наповнення тару з відходами закривають герметично сталевий кришкою.

- Відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин.

- Відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення.

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» [34] повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів.

4.7 Висновки до розділу 4

В результаті проведеної роботи було проаналізовано умови праці, шкідливі та небезпечні фактори, з якими стикається працівник. Параметри та певні характеристики приміщень були визначені для роботи над запропонованим проектом, з описом того, які заходи повинні бути зроблені для того, щоб приміщення відповідало необхідним стандартам і було зручним і безпечним для працівника. Наведено рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електричної безпеки. Схема, розміри приміщення та значення температури, вологості та рухливості повітря, необхідна кількість і потужність ламп та інших параметрів, значення яких впливають на умови праці робітника, а також - дані інструкції з охорони праці, безпеки при роботі на комп'ютері.

Він також визначає основні екологічні аспекти впливу на навколишнє середовище та заходи, вжиті для їх подолання.

ВИСНОВКИ

У магістерській роботі проведено огляд існуючих технологій для передачі інформації між двома вузлами мережі. Були розглянуті як однорангові, так і багаторангові мережі. Було визначено деякі проблеми передачі мультимедійного контенту.

Під час досліджень, проведених в першому розділі даної роботи, були виділені переваги та недоліки різних протоколів та технологій для передачі інформації між пристроями на різних платформах.

Далі було проведено огляд, аналіз, дослідження та порівняння існуючих технологій для передачі інформації між двома вузлами мережі. А саме java, flash та WebRTC. Найбільш підходящою виявилась технологія передачі аудіо та відео контенту WebRTC. Було розглянуто основні алгоритми технології для роботи з відео та аудіо.

Також було запропоновано метод каскадування для спрощення масштабування та поліпшення якості передачі контенту через WebRTC.

Потім було побудовано базовий WebRTC додаток, розглянуто роботу його компонентів, таких як ICE-кандидати, STUN та TURN сервери, та методи тестування, а саме: локальне та зовнішнє налагодження. До локального можна віднести розбір SDP протоколу та панель розробника internals в браузері Google Chrome. До зовнішнього тестування можно віднести різні аналізатори мережевих пакетів, наприклад, Wireshark, який дозволяє відстежити стан пакетів, що передаються.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Jimmy To T. Interactive Video-On-Demand Systems: Resource Management and Scheduling Strategies / T. Jimmy To, B. Hamidzadeh., 2012.

2. Lawson B. Introducing HTML5 / B. Lawson, R. Sharp., 2011.

3. Спецификация протокола BitTorrent. Режим доступу: www.URL: http://www.bittorrent.org/beps/ bep_0003.html

4. «Эволюция Интернета». Режим доступу: www.URL: http://www.evolutionoftheweb.com/

5. Real-Time Communication with WebRTC. Сальваторе Лорето. 2014г. 164с

6. Стандарт IETF о peer-to-peer сетях. Режим доступу: www.URL: https://tools.ietf.org/html/rfc5694

7. Java. Режим доступу: www.URL: https://en.wikipedia.org/wiki/Java

8. Патрик Нотон, Герберт Шилдт Полный справочник по Java, Издательство "Диалектика",2007

9. Джуди Бишоп - Эффективная работа Java., 2002

10. James Gosling, Bill Joy, Guy Steele, Gilad Bracha - The Java Language Specification, Second Edition

11. Tim Lindholm, Frank Yellin - The Java Virtual Machine Specification, Second Edition

12. Гослинг Дж., Арнольд К. - Язык программирования Java

13. .Тверезовский Д. И. Macromedia Flash MX 2004. Самоучитель., 2005.

14. Flash and rtmp. Режим доступу: www.URL: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C0405A65844B6B73C9B195FA1596B14C?doi=10.1.1.177.3489&rep=rep1&type=pdf

15. Adobe Real Time Messaging Protocol RTMP. Режим доступу: www.URL: http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol

16. Adobe Real Time Messaging Protocol. Режим доступу: www.URL: http://www.adobe.com/devnet/rtmp/

17. Описание технологии WebRTC. Режим доступу: www.URL: http://www.3cx.ru/webrtc/

18. Технология WebRTC: аудио- и видеочат в браузере. Режим доступу: www.URL: http://ichip.ru/tekhnologiya-webrtc-audio-i-videochat-v-brauzere.html

19. Всё о WebRTC. Режим доступу: www.URL: http://blog.trueconf.ru/reviews/webrtc.html

20. Real-Time Communication with WebRTC. Сальваторе Лорето. 2014г. 164с

21. Tarakanov A N , Nazarovsky A E , Moseev A L Improvement of NLMS adaptive algorithm performance in double talk mode // 2nd IEEE international conference on circuits and systems for communications Moscow, 2004

22. An Efficient Audio Noise Filter by F. J. Archibald, submitted for publication.

23. WebRTC Media & Broadcasting Server. Режим доступу: www.URL: https://flashphoner.com/webrtc-serverpotokovogo-video-dlya-onlajjn-t/?lang=ru

24. Видеосвязь через браузер: Flash или WebRTC // Видеосвязь через браузер. Режим доступу: www.URL: https://trueconf.ru/blog/reviews/videosvyaz-cherez-brauzer-flashili-webrtc.html

25. WebRTC. Видеоконференции в браузере // WebRTC. Видеоконференции в браузере. Режим доступу: www.URL: https://trueconf.ru/webrtc.html

26. Инструкция создания WebRTC приложения // Инструкция создания WebRTC приложения. Режим доступу: www.URL: https://habrahabr.ru/post/198632/

27. Что такое STUN-сервер? www.URL: Режим доступу: https://www.3cx.ru/voip-sip/stun-server/

28. Что такое Traversal Using Relay NAT сервер ? Режим доступу: www.URL: https://ru.wikipedia.org/wiki/Traversal_Using_Relay_NAT

29. ДСТУ Б А.3.2-13: 2011 «Будівництво. Електробезпека. Загальні вимоги» - Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/z0072-13/paran21

30. ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Прийнято та надано чинності наказом Мінрегіонбуду України від 15.06.2016 р. № 158 з 01.01.2017 р. Режим доступу: www.URL: https://dbn.co.ua/load/normativy/dstu/dstu_b_v_1_1_36/5-1-0-1759

31. НПАОП 0.00-1.04-07 «Про затвердження Правил вибору та застосування засобів індивідуального захисту органів дихання» - https://zakon.rada.gov.ua/laws/show/z0285-08

32. ГОСТ 12.1.004-91 ССБТ «Пожежна безпека. Загальні вимоги». Затверджено і введено в дію Ухвалою Державного комітету СРСР з управління якістю продукції та станд артів від 14.06.91 №875. Режим доступу: www.URL: http://docs.cntd.ru/document/9051953

33. ДБН В.2.5-28-2018 «Природнє і штучне освітлення» - Режим доступу: www. URL: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf

34. Закон України «Про відходи». Відомості Верховної Ради України (ВВР), 1998, № 36-37, ст.242. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/187/98-вр

Додаток А

Слайди комп'ютерної презентації

Рисунок А.1 – Титульний лист

Рисунок А.2 – Основні моделі мережевих взаємодій

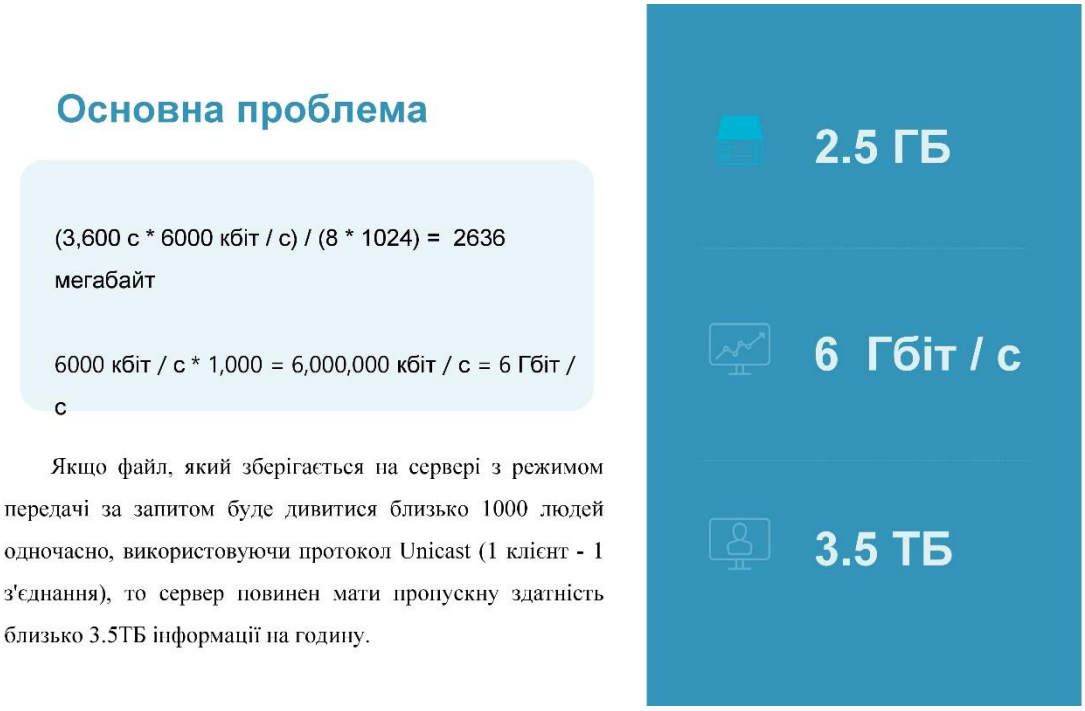


Рисунок А.3 – Основна проблема

Рисунок А.4 – Моделі передачі контенту

Рисунок А.5 – Три основні технології

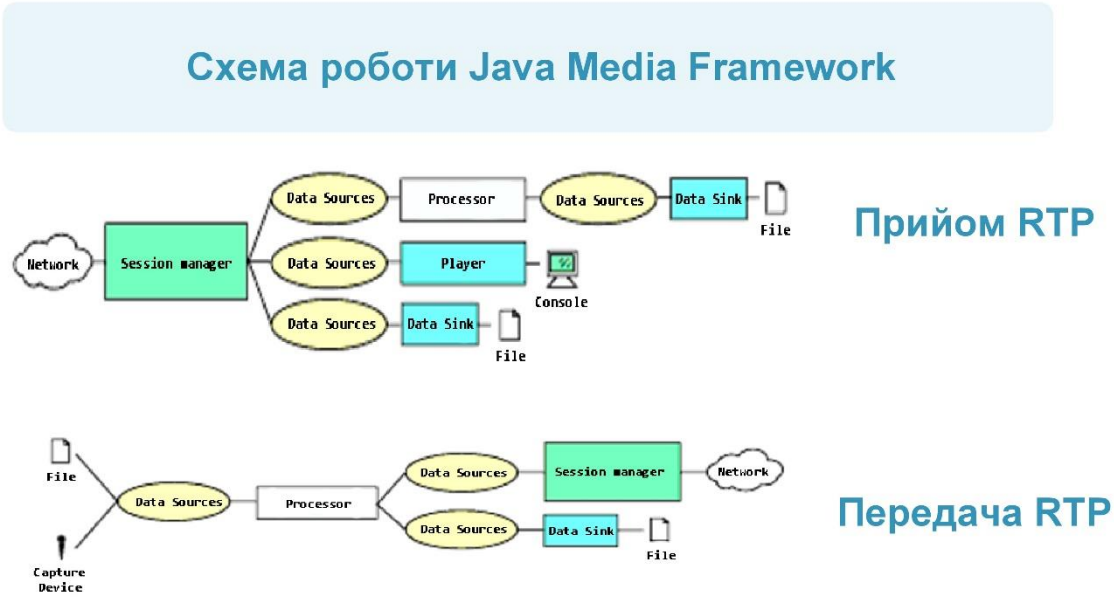


Рисунок А.6 – Схема роботи JMF

Схема роботи Flash Player

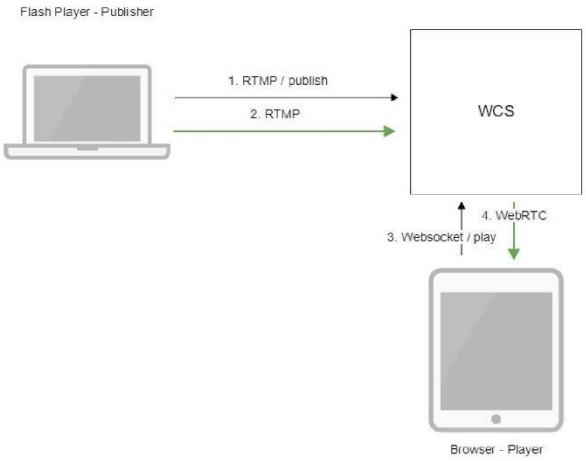


Рисунок А.7 – Схема роботи Flash Player

Схема роботи WebRTC

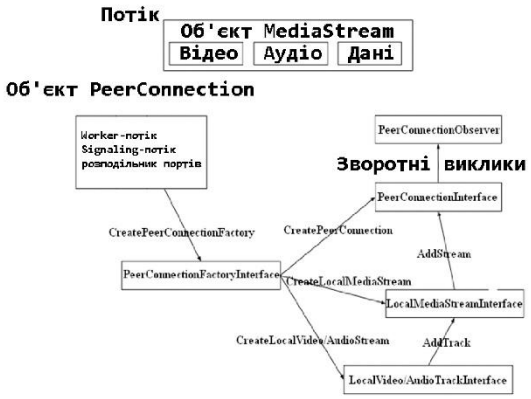


Рисунок А.8 – Схема роботи WebRTC

Системні вимоги			
Технологія	FLASH (RTMP)	JMF (RTP)	WebRTC
Операційна система	Linux CentOS 6	Solaris/Linux	Linux Debian 8, 9
Хмарні рішення	AWS, Google Cloud, Azure	-	AWS, Google Cloud, Azure
Версія	1.3.0 EOL	2.1.1e	1.0
CPU	> 2 ядра OpenGL	> 2 ядра	> 2 ядра
RAM	> 2 Gb	> 2 Gb	> 4 Gb
HDD/SDD	> 4 Gb	> 10 Gb	> 10 Gb
Основні характеристики			
SPLIT модуль	Так	Ні	Так
Відеоканали на одиницю	1-120	1-10	1-240
Web Plugin	Flash Player	Java	Не потребується
SIP Connect	Version 11	Version 4.6	Version 13
Пріоритет голосу над відео	Так	-	Так
Mobile SDK	Не підтримується	Android	iOS & Android
API Framework	Flash SWF Editor	Java SE	VideoRTC.js
Веб панель	Не підтримується	Для дебагу та моніторингу	Для дебагу та моніторингу

Рисунок А.9 – Таблиця порівняння №1

Протоколи зв'язку			
Протокол	Adobe's Open Public Protocol	RTSP	ETF Open Standard Protocol
Аудіо кодек	G711, Speex	G711, A-law, IMA4 ADPCM	G711, Opus
Відео кодек	H263 Sorenson	Cinepak, MJPEG, RGB, YUV, VCM, H.261, H.263	VP8, VP9, H264
IP-пакет	TCP	UDP	UDP
Формат експорту відео	Не підтримується	.mpeg2 .mpeg4	.mweb .mp4
Автовиклик	Так	Ні	Так
Автологін	Так	Так	Так
Підтримка веб-браузерів			
Chrome	FLASH заблоковано	Заблоковано	Підтримується
Firefox	FLASH заблоковано	Заблоковано	Підтримується
MS Internet Explorer	Підтримується	Заблоковано	Не підтримується
MS Edge	FLASH заблоковано	Заблоковано	Не підтримується з 03-2019
Opera	Підтримується	Заблоковано	Підтримується
Safari	FLASH заблоковано	Заблоковано	Тільки звук

Рисунок А.10 – Таблиця порівняння №2

Скасування акустичного відлуння

LMS-алгоритм використовується в додатках, де апріорі можна оцінити гранично допустиме значення кроку збіжності. Така оцінка можлива, якщо відомі статистичні характеристики вхідного сигналах.

У додатках, де ці характеристики невідомі або змінюються з часом (наприклад, при обробці нестаціонарних сигналів), часто використовується нормалізований LMS-алгоритм (Normalized LMS, NLMS), що є однією з різновидів LMS-алгоритму.

$$X_N(0) = 0_N, h_{N_x}(0) = 0_N$$

$$X_{N_m}(k)|2, \dots, N_m = X_{N_m}(k-1)|1, \dots, N_m - 1, X_{N_m}(k)|1 = x_m(k), m = 1:M$$

$$X_N(k) = \left[X_{N_1}^T(k), X_{N_2}^T(k), \dots, X_{N_m}^T(k), \dots, X_{N_M}^T(k)\right]$$

$$a_{N_x}(k) = d(k) - h_{N_x}^H(k-1)x_N(k)$$

$$h_{N_x}(k) = h_{N_x}(k-1) + \frac{\mu}{x_N^H(k)x_N(k)+\delta^2} x_N(k)a_{N_x}^*(k)$$

Рисунок А.11 – Скасування акустичного відлуння

Робота jitter - буферу

Логіка роботи джиттер-буфера кінцевих абонентських пристроїв полягає в компенсації джиттера наскрізної затримки передачі пакетів шляхом їх буферизації протягом часу, достатнього для послідовного і своєчасного відтворення аудіо- та відеоінформації, як це було сформовано джерелом.

$$x_{qpl}(t) = \beta \max\left(\widehat{\mathrm{JD}}[k], 0\right) \widehat{\mathrm{R}}[k]$$

Рисунок А.12 – Робота jitter - буферу

Оцінка коефіцієнтів придушення шуму

Оцінка амплітудного спектра шуму [22] |N(k,m)| формується шляхом усереднення амплітуд попередніх відліків сигналу:

$$|\hat{N}(k,m)| = \tilde{a}_d(k,m-1)|\hat{N}(k,m-1)| + (1-\tilde{a}_d(k,m-1))|\bar{X}(k,m)|$$

де $\tilde{a}_d(k,m) = \alpha_d + (1-\alpha_d)\hat{p}(k,m)$, це параметр згладжування, що змінюється в часі, який залежить від умовної ймовірності присутності мовного сигналу $\hat{p}(k,m), \alpha_d$ - параметр згладжування, який визначає час усереднення, коли мова відсутня ($0 < \alpha_d < 1$).

$$G_{NR}(k,m) = max\left\{\sqrt{|\bar{X}(k,m)|^2 - v|\hat{N}(k,m)|^2|\bar{X}(k,m)|^2}, 10^{-RL/20}\right\}$$

Де v - коефіцієнт вирахування (1< v < 6), RL - параметр, що визначає бажаний рівень залишкового шуму.

Рисунок А.13 – Оцінка коефіцієнтів придушення шуму

Схема реалізації каскадування SFU

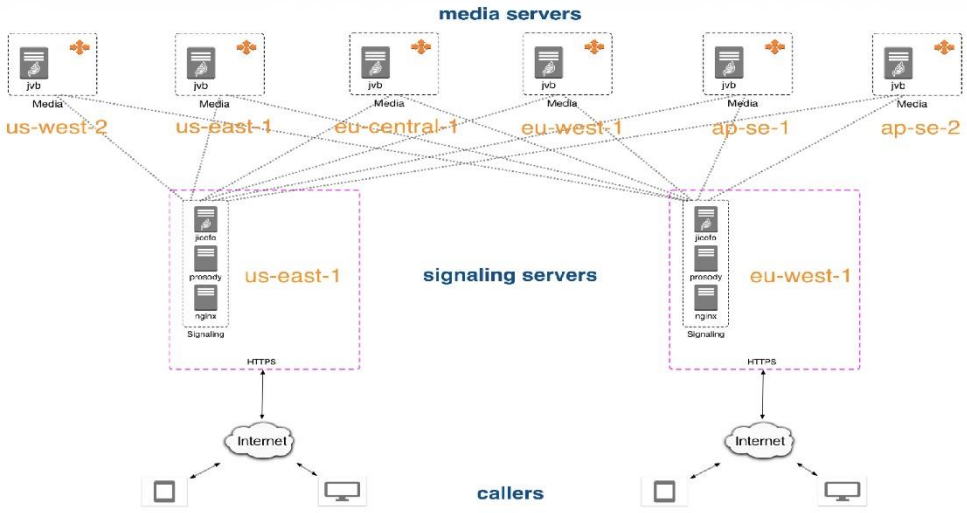


Рисунок А.14 – Схема реалізації каскадування SFU

Результати каскадування

У з'єднанні SFU є свої плюси і мінуси. З одного боку, в описуваній ситуації, час обміну між учасниками стає більше за інших нових прижків по мережах. З іншого боку, можливість зменшити цей час, коли ми говоримо про зв'язок «клієнт» - «перший сервер», тоді ми будемо слухати медіапоток з меншою затримкою.

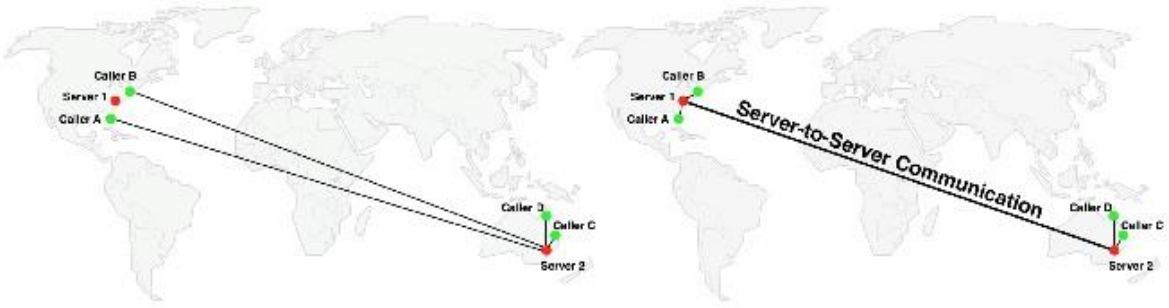


Рисунок А.15 – Результати каскадування

Тестування кросплатформності веб – додатку

Браузер	Версія	Операційна система
chrome	79.0.3945.117	windows 10
chrome	79.0.3945.117	windows 7
firefox	69.0	windows 10
chrome	74.0.3729.136	Android 9
yandex browser	19.12.1.229	windows 10
opera	65.0.3467.78	windows 10
safari	13.0.4	IOS 13
firefox	67.0	ubuntu

Рисунок А.16 – Тестування кросплатформності веб - додатку

Таблиця підтримуваних браузерів

Браузер	Версія	Браузер	Версія
IE	Не підтримується	Chrome for Android	79
Edge	76	Firefox for Android	68
Firefox	22	UC browser for Android	Не підтримується
Chrome	23	Samsung Internet	4
Safari	11	QQ Browser	1.2
Opera	18	Opera Mini	46
IOS Safari	11		

Рисунок А.17 – Таблиці підтримуваних браузерів

Висновки

1. Порівняно та визначено підходящу технологію для кросплатформного обміну інформацією.

2. Розроблено та впроваджено кросплатформний веб – додаток з методом каскадування SFU.

Рисунок А.19 – Висновки