Санкт–Петербургский государственный университет

Тарасов Даниил Антонович

Научно-исследовательская работа

Исследование систем видеоконференций

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа СВ.5003.2021 «Программирование и информационные технологии»

Научный руководитель:
доцент, кафедра компьютерного моделирования и многопроцессорных систем, к.ф. - м.н. Корхов Владимир Владиславович

Санкт-Петербург
2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время системы видеоконференций являются неотъемлемой частью повседневной коммуникации. Развитие технологии связи привели к появлению различных платформ и технологии, обеспечивающих возможность удаленного общения. Среди них немаловажное место занимает фреймворк WebRTC (Web Real-Time Communication), который позволяет осуществлять передачу аудио и видеоданных в реальном времени через веб-браузер без необходимости установки дополнительного программного обеспечения.

В рамках данной научной исследовательской работы мы рассмотрим историю развития систем видеоконференций, обоснуем выбор WebRTC в контексте современных требований и ожиданий пользователей. Кроме того, мы рассмотрим основные проблемы, с которыми сталкиваются разработчики систем видеоконференций и варианты их решения.

Глава 1. История

История систем видеоконференций началась в 1920-ых годах, когда появились первые стабильные и работающие телекамеры, тогда же компания ATT Bell Telephone Laboratories создала работающий комплекс телефонной связи, которые транслировал изображение на расстояние 200 миль. В 1930-ых та же компания ATT продемонстрировала сеанс двусторонней видеосвязи между офисами ATT на Манхэттене, однако затянувшиеся последствия Великой депрессии затормозили развитие видеосвязи.

В 1980-ых компания Compression Labs составила конкуренцию ATT и выпутила CLI T1 в качестве первой коммерческой системы групповой видеоконфенцсвязи. Ее первоначальная стоимость составляла 250000 долларов, а каждый звонок стоил 1000 долларов в час.

В 1990-ых произошел Бум Интернета и развития цифровой телефонии. В 1991 году студенты факультета компьютерных наук Кембриджского университета изобрели первую веб-камеру.

В начале 2000-ых появились смартфоны, оснащенные камерами на задней и передней панелях для съемки фотографий. Три эстонских инженера-программиста представили Skype в августе 2003 года. Два бывших сотрудника Yahoo основали WhatsApp в 2009 году как приложение для мгновенного обмена сообщениями. В 2020-ых блокировка COVID-19 заставило многих людей по всему миру работать из дома, что привело к буму на все видеоустройства и большой популярности систем видеоконференций [1].

Глава 2. Почему WebRTC?

Инициатива WebRTC достигла впечатляющих результатов с точки зрения промышленного интереса, проникновения технологии в устройства конечных пользователей и постоянно растущего сообщества разработчиков. Сегодня WebRTC поддерживается основными мобильными платформами и интернет-браузерами, позволяя потенциально миллиардам пользователей беспрепятственно устанавливать сеансы связи в режиме реального времени [2].

Кратко рассмотрим ключевые особенности WebRTC:

– Открытый стандарт;

– Разные реализации;

– Доступность в браузерах;

– Обязательное шифрование;

– Отображение NAT (NAT Traversal);

– Перепрофилирование существующих технологий;

– Контроль перегрузки (congestion control);

– Низкая задержка (на уровне долей секунды, sub-second latency) [3].

Глава 3. WebRTC

Рассмотрим WebRTC более подробно. WebRTC (Web Real-Time Communication – коммуникация в режиме реального времени) – это набор правил, позволяющий двум агентам вести двунаправленную безопасную коммуникацию в режиме реального времени.

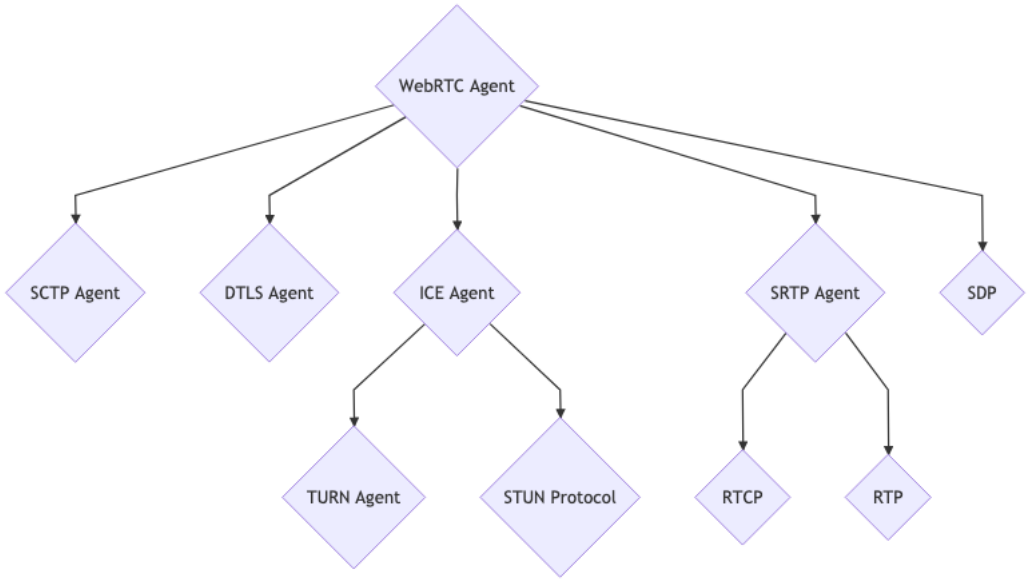

Рисунок 1 – WebRTC - оркестратор

Фактически WebRTC – это оркестратор нескольких других протоколов, Рисунок 1.

В процессе установки соединения можно выделить 4 основных этапа:

– Сигнализация;

– Подключение;

– Безопасность;

– Коммуникация [3].

Далее рассмотрим эти 4 этапа.

3.1. Сигнализация

Сигнализация – это подготовка к совершению звонка. После обмена необходимой информацией участники могут общаться друг с другом напрямую.

Для совершения этого этапа WebRTC используется протокол SDP. Он позволяет двум участникам обменяться состоянием, необходимым для установки соединения. Протокол описания сессии определен в RFC 8866. Описание сессии представляет собой описание медиа (media description) и состоит из пар ключ/значение [21].

3.2. Подключение

В WebRTC используется P2P (Peer-to-Peer) архитектура. Задача WebRTC обеспечить возможность двунаправленной коммуникации между двумя участниками.

ICE – это протокол, который определяет наилучший способ для установки соединения между двумя участниками. Каждый участник определяет путь, по которому он может быть достигнут, затем ICE выбирает наиболее подходящую пару путей [22].

3.3. Безопасность

Каждое соединения WebRTC аутентифицируется и шифруется. Таким образом, мы можем быть уверены, что третья сторона не видит того, что мы отправляем, и не может добавлять свои сообщения.

Для обеспечения безопасности соединения WebRTC использует DTLS и SRTP протоколы.

DTLS позволяет выполнить подготовку сессии и безопасно обмениваться данными между пирами. DTLS – это клиент-серверный протокол, поэтому одна из сторон должна инициировать рукопожатие, в ходе него каждая из сторон генерирует сертификат. После завершения рукопожатия каждый сертификат сравнивается с его хешем, содержащим описание сессии, все это позволяет убедиться, что рукопожатие произошло с ожидаемым участником.

SRTP был разработан для безопасного обмена медиаданными. Для создания сессии SRTP мы инициализируем ее с помощью ключей, сгенерированных DTLS. После установки соединения стороны могут обмениваться зашифрованными медиаданными [23].

3.4. Коммуникация

Для обмена данными WebRTC предоставляет каналы. Между двумя участниками может быть 65534 каналов. По умолчанию он гарантирует сохранение порядка сообщений.

Канал данных – это абстракция потоков, из которых состоит SCTP. SCTP – это транспортный протокол, являющийся альтернативой TCP и UDP, он определен в RFC 4960. Все настройки, связанные с продолжительностью существования канала и порядком доставки сообщений, передаются агенту SCTP [24].

Глава 4. Ожидания конечных пользователей

Опишем основные ожидания конечных пользователей систем видеоконференций:

– Большое количество участников в видеоконференции;

– Быстрая установка соединения между участниками звонка;

– Низкие задержки между звонящими;

– Высокое качество аудио и видео в звонке [4].

Далее расмотрим проблемы, которые идут вразрез ожиданиям конечных пользователей.

Глава 5. Основные проблемы, влияющие на качество видеоконференций

В современном мире на уровне сетевых коммуникаций возникают различные проблемы, влияющие на качество и надежность передачи данных, которые в конечном итоге могут негативно сказываться на качестве видеоконференций:

– Перегрузка сетей;

– Потеря пакетов;

– Пропускная способность сетей;

– Jitter;

– NAT;

– MTU;

– Reordering [4].

Технология WebRTC представляет мощный набор инструментов, который способен решить эти проблемы, обеспечивая стабильное взаимодействие в реальном времени.

5.1. Перегрузка сетей

Одна из распространненых сетевых проблем – перегрузка сетей, появляется, когда сетевой узел или линия связи переносит больше данных, чем может обрабатывать. Частые эффекты включают задержку в очереди, потерю пакетов или блокировку новых соединений, в худшем случае это может привести к снижению пропускной способности сети.

Для отслеживания и предотвращения перегрузки сетей существуют разные решения, например:

– Экспоненциальная выдержка – это алгоритм, использующий обратную связь для мультипликативного уменьшения частоты некоторого процесса, чтобы постепенно, чтобы постепенно найти приемлимую частоту. Этот алгоритм обычно используется для планирования повторных отправок после коллизий. После первой коллизии каждый отправитель будет ждать 0 или 1 slot time. После второй коллизии отправители будут ждать где-то от 0 до 3 slot times включительно. По мере увеличения количества попыток повторной отправки число вариантов для задержки растет экспоненциально;

– Приоритезация сетевых пакетов – это метод, в котором пакеты, помеченные как "важные пропускаются в первую очередь, а менее важные придерживаются, пока линия не освободится [11].

5.2. Потеря пакетов

Потеря пакетов это частая проблема, которая характеризуется тем, что сообщения теряются при передаче, то есть не доходят до адресата. Для решения этой проблемы WebRTC предлагает несколько вариантов:

– Если были утеряны медиа-пакеты, то мы можем скрыть этот факт и восстановить пакеты самостоятельно, "догадываясь"о том, что было потеряно. Некоторые кодеки умеют восстанавливать потерянные кадры заменой их на тишину или путем повторения уже принятой речи, например, последнего кадра [8];

– Retransmission (Ретрансляция) в случае, когда нас устраивают возникающие задержки, необходимые для запроса повторной отправки. Осуществляется RTP [9];

– FEC (Forward Error Connection) – это механизм, при котором медиапакета заранее дублируются и отправляются по сети несколько раз. Таким образом, даже если некоторые пакеты не будут получены, медиапоток все равно можно будет правильно разобрать и декодировать [10].

5.3. Пропускная способность сетей

Начнем с определения пропускной способности сети – это количество данных, которые можно передать по сети. Это динамическая величина, которая зависит от нагрузки, то есть от количества людей, использующих этот маршрут. Перегрузка сети в худшем случае может привести к потери пакетов, увеличению задержки и джиттера, что негативно сказывается на коммуникации в реальном времени.

Подход WebRTC заключается в том, чтобы попытаться оценить доступную пропускную способность сети и ограничить отправителя от отправки черезчур большего от нашей оценки. Оценка пропускной способности основана на эвристике, которая моделирует поведение сети и пытается его предугадать.

В WebRTC используются два основных подхода для оценки пропускной способности [5]:

– REMB [6];

– transport-cc [7].

5.4. Jitter

При отправке пакетов нет гарантии, что они будут доставлены в те же промежутки, с которыми были отправлены. Разница от ожидаемого интервала получения пакетов называется джиттером. Чем он выше, тем сильнее он влияет на качество передачи данных.

В виду того, что голос и видео чувствительны ко времени, при получении медиапакетов их необходимо собрать, переупорядочить и затем определять время, основываясь на последовательности и различиях, в которых они были сгенерированы, а не на последовательности и времени, в которых они были получены. Этим занимается буффер джиттера и в WebRTC имеется собственная реализация, которая учитывает задержку сети, любые наблюдаемые потери пакетов и "расстояние"между входящими аудио- и видеопакетами [12].

Кратко алгоритм можно описать так, Рисунок 2:

1. Каждый пакет добавляется в буфер джиттера сразу после его получения;

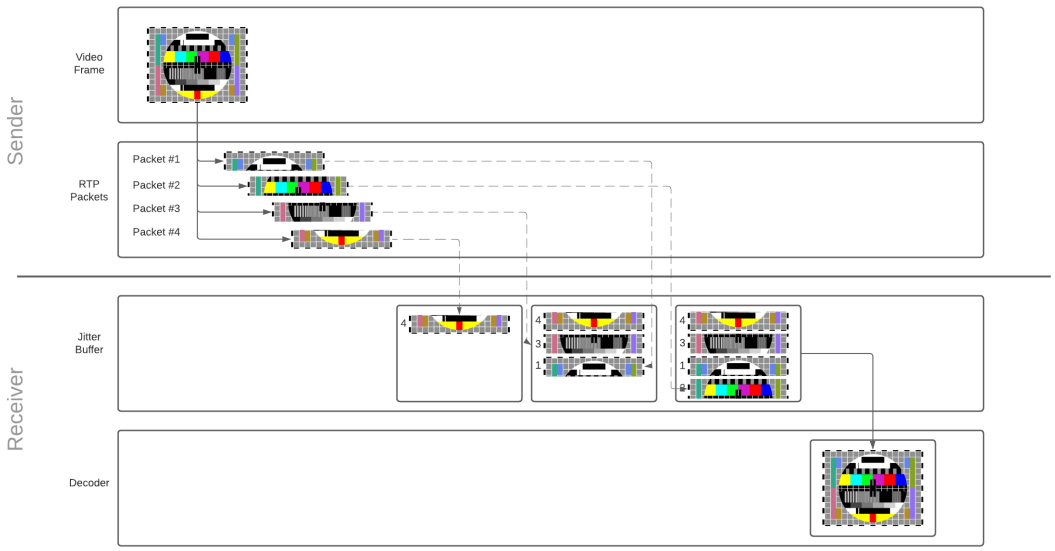

Рисунок 2 – Джиттер буффер

2. Как только пакетов становится достаточно для реконструкции кадра, пакеты, составляющие кадр, освобождаются из буфера и передаются для декодирования;

3. Декодер, в свою очередь, декодирует и отрисовывает видеокадр на экране пользователя.

Поскольку буфер джиттера имеет ограниченную емкость, пакеты, которые остаются в буфере слишком долго, отбрасываются [13].

5.5. NAT

NAT расшифровывается как трансляция сетевых адресов, обычно располагается между частной и публичной сетями и встроена в устройства сетевой маршрутизации. Входящий трафик в частную сеть направляется через привязку публичного адреса, которая возникает на устройстве NAT.

WebRTC должен иметь возможность передевать медиа между двумя абонентами, которые могут находиться за NAT устройствами, для этого необходимо, чтобы внешние пакеты могли проходить во внутреннюю сеть. STUN (Session Traversal Utilities for NAT) – это стандартный метод обхода NAT, используемый в WebRTC.

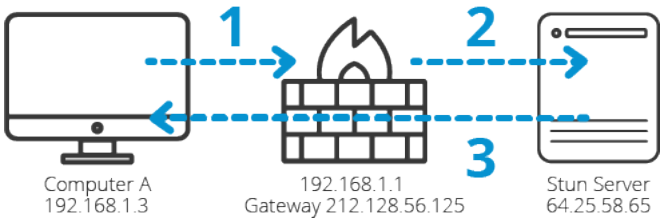

Рисунок 3 – STUN

STUN-сервер позволяет своим клиентам находить публичный адрес, тип NAT, за которым они находятся и порт Интернета, связываемый NAT с конкретным локальным портом, Рисунок 3. Затем эта информация используется WebRTC для настройки связи UDP между клиентами. Протокол STUN определяется стандартом RFC 3489.

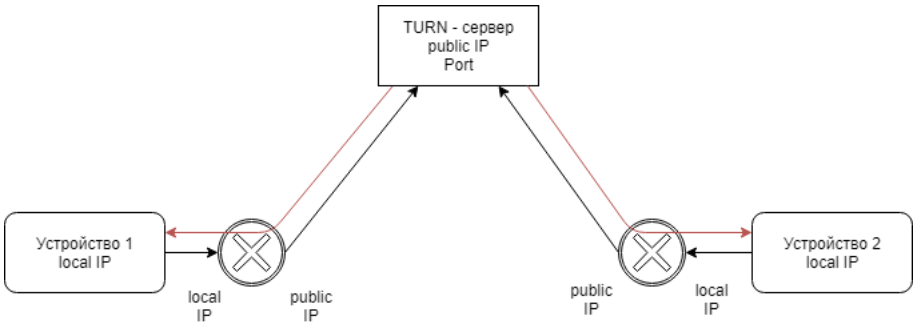

Рисунок 4 – TURN

TURN (Traversal Using Relays around NAT) используется для передачи мультимедиа через сервер TURN, когда использование STUN невозможно. Решение о том, использовать STUN или TURN, принимается протоколом ICE. TURN-сервер фактически выступает проксирующим звеном между двумя клиентами, пропуская через себя весь трафик, Рисунок 4.

ICE (Interactive Connection Establishment) – это еще одно дополнение к протоколам STUN и TURN. Его задачи:

– Собрать данные об интерфейсах;

– Проверить удаленные сервера STUN;

– Проверить удаленные сервера TURN;

– Проверить возможность установления соединения.

ICE выбирает самый легкий для прохождения маршрут (процедура номинирования пар) и выполняет проверку доступности между пиром и клиентом (могут ли они достучаться друг до друга). Приоритезация такая: без STUN сервера – самый высокий приоритет, с использованием STUN – пониже, с TURN – самый низкий.

5.6. MTU

MTU (Maximum Transmission Unit) – это максимальный размер пакета, который может быть передан по сети. Если пакет передается с MTU большим, чем максимальный MTU на сети, то этот пакет фрагментируется. Конечно, он потом соберется обратно, однако если потеряется какая-то часть пакета, то потеряются все пакеты, поэтому необходимо оптимально работать с таким размером пакета, который соответствует MTU сети [4].

WebRTC устанавливает размер MTU около 1200 байт и использует его для своих расчетов пакетирования (плюс-минус несколько байт). Использование такого значения гарантирует, что WebRTC будет хорошо работать в большинстве сетевых конфигураций [14].

5.7. Reordering

Во время путешествия сетевых пакетов между источником и пунктом назначения есть вероятность переупорядочивания пакетов, то есть пакеты могут прибыть в другом порядке относительного того, в каком они были отправлены источником.

Для приложений видеоконференций, использующий протокол UDP и не использующих, например, метод коррекции ошибок вперед (FEC), переупорядочивание пакетов является проблемой, поскольку обычно приводит к потере данных, поскольку пакеты, приходящие не по порядку, отбрасываются, в худшем случае можно потерять целый видеокадр или аудиосегмент.

Один из простейших методов коррекции переупорядочивания пакетов заключается в наличии памяти для неупорядоченных пакетов, в которой хранятся все пакеты, полученные с неожиданным порядковым номером. Эта

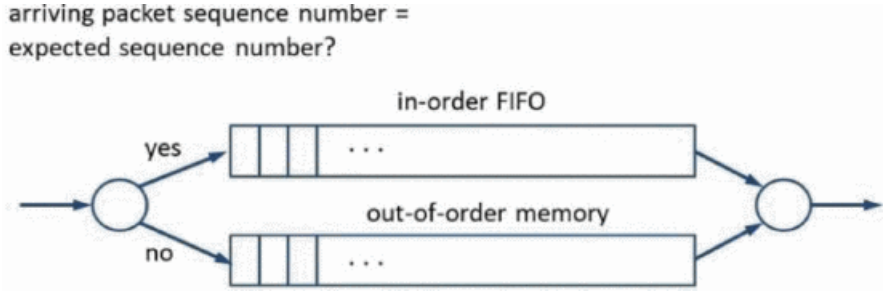

Рисунок 5 – Упорядочивающая архитектура с одной out-of-order памятью

память затем проверяется, когда ожидаемый номер не присутствует в конце очереди FIFO. Идея проиллюстрирована на Рисунке 5. Out-of-order буфер может быть организован как FIFO для простой коррекции переупорядочивания [15].

Глава 6. Передача видео

Еще одна из важнейших проблем, решаемых WebRTC – передача видео. Это сложный процесс, для хранения 30-минутного несжатого 720 8-битного видео требуется около 110 Гб. В таких условиях конференция с четырьмя участниками является невозможной. Для решения этой проблемы используют сжатие.

Сжатие делится на 2 типа:

1. Сжатие внутри кадра (intra-frame compression) – это сжатие, которое уменьшает количество бит, используемых для описания единичного видеофрейма. Подобная техника используется для сжатия неподвижных изображений, например, JPEG;

2. Межкадровое сжатие (inter-frame compression) – способ не передавать одинаковую информацию дважды.

   Кадры при межкадровом сжатии делятся на 3 типа:

– I-Frame – полное изображение, которое может быть декодировано без каких-либо изменений;

– P-Frame – частичное изображение, содержащее только изменения предыдущего изображения;

– B-Frame – частичное изображение, представляющее собой модификацию предыдущего и последующего изображений.

Ниже приведена визуализация трех типов кадров, Рисунок 6.

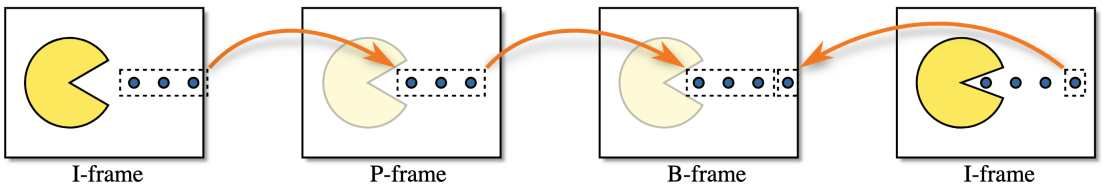

Рисунок 6 – Типы кадров

6.1. Потеря изображения

Для решения проблемы с потерей изображения используются сообщения FIR (Full Intra Request) и PLI (Picture Loss Indication). Эти сообщения запрашивают у отправителя полный ключевой кадр.

PLI используется, когда декодер получает частичные кадры и не модет их декодировать. Такое может произойти при потере данных или ошибки декодера.

FIR не должен использоваться при потере пакетов или кадров согласно RFC 5104. FIR запрашивает ключевой кадр, например, при подключении к сессии нового участника, так как для начала декодирования видео требуется ключевой кадр, до его получения декодер будет отклонять остальные кадры [16].

Глава 7. Масштабирование участников видеоконференций

Хотя WebRTC представляет собой мощный интрумент для решения многих проблем при создании системы видеоконференций – он не идеален. Фреймворк WebRTC был разработан с учетом одноранговой (P2P) архитектуры, такой подход имеет проблемы с масштабируюмостью, если мы хотим одновременное присутствие большого количества участников. В WebRTC общающиеся стороны должны кодировать отдельный поток для получателей. Таким образом, каждый получатель связан независимым и выделенным кодером на стороне отправителя. Для поддержки N участников конференции с помощью чистой Mesh сети потребуется N * (N – 1) / 2 каналов. То есть требования к пропускной способности устройств будет расти квадратично по отношению к числу участников видеоконференции, Рисунок 7 [17].

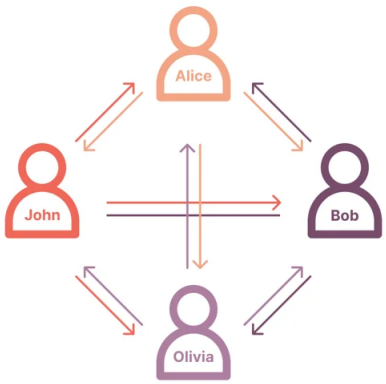

Рисунок 7 – Mesh архитектура

Чтобы улучшить масштабируемость этой архитектуры, можно использовать промежуточный медиасервер. В WebRTC есть 2 распространенных подхода – использование устройства многоточечной конференции (MCU) или устройства селективной переадресации (SFU).

7.1. SFU

В WebRTC функции контроллера конференции может выполнять устройство селективной переадресации (SFU), задачей которого является получение всех потоков и принятие решения о том, какой поток должен быть отправлен какому участнику. Задача контроллера – оптимизация доставки потоков реального времени от отправителя к получателю.

В отличие от подхода MCU, SFU не требует декодирования/кодирования и поэтому является более легким. Его основная задача это принимать все потоки от участников и выборочно пересылать один или несколько потоков каждому получателю, Рисунок 8.

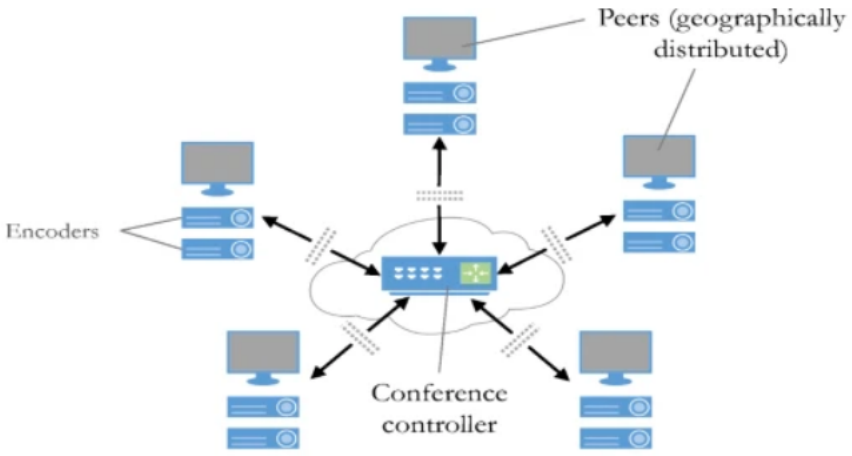

Рисунок 8 – SFU архитектура

При большом количестве участников количество пересылаемых потоков необходимо ограничивать, чтобы не тратить пропускную способность. По этой причине Грозева и др. разработали алгоритм идентификации докладчиков, который определит N последних доминирующих докладчиков конференции. Для экономии полосы пропускания только эти N потоков передаются участникам видеоконференции.

По сути контроллер выполняет 2 основные задачи:

1. Получает все кодированные потоки от отправителя и динамически направляет их получателям, исходя из доступной пропускной способности, Рисунок 9;

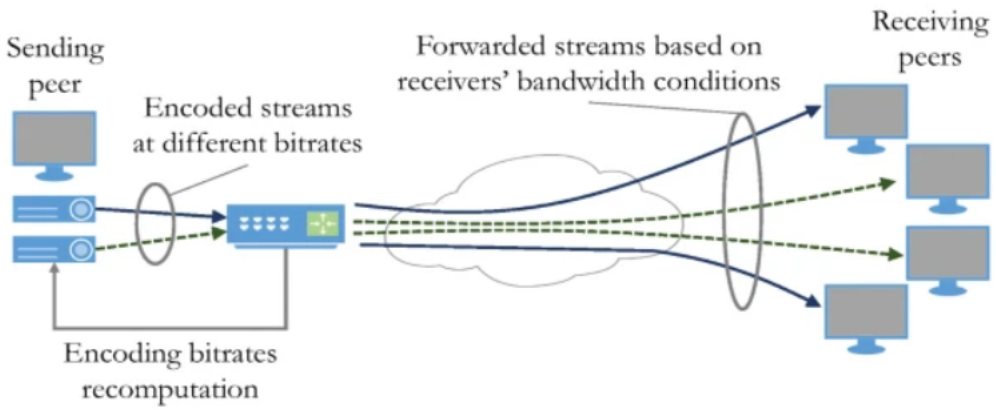

Рисунок 9 – SFU controller dynamic forwarding

2. Переодически пересчитывает битрейты кодирования отправителя, чтобы лучше следовать долгосрочным колебаниям сети получателя [18].

7.2. MCU

Подход MCU заключается в том, чтобы получить все потоки от участников, декодировать и компилировать их в один общий поток, который отправляется обратно участникам. В итоге каждый участник должен отправлять и получать только один поток. Стоит понимать, что операции MCU дорогостоящие и требуют больших вычислительных затрат из процессов декодирования-смешивания-кодирования.

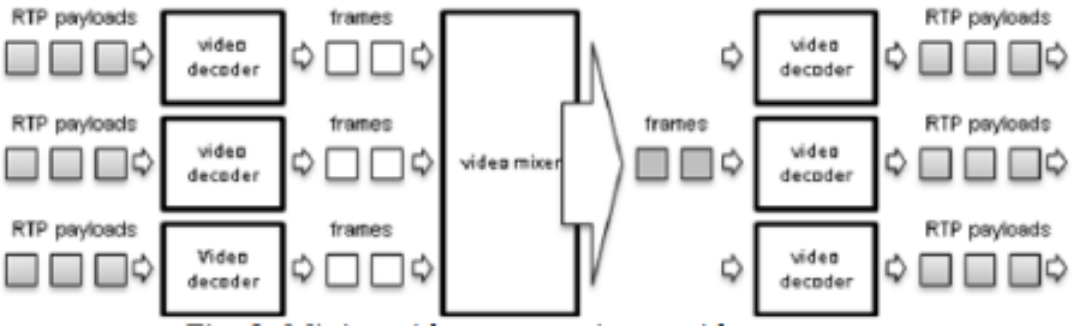

Рисунок 10 – MCU смешивание аудиопотоков

MCU назначает аудио/видео кодер и декодер, которые способны декодировать RTP-потоки от участников, кодер кодирует смешанные видеокадры

обратно в полезную нагрузку RTP для каждого участника. Простое смешивание аудиопотоков всех участников в один поток привело бы к появлению эха от голоса. Чтобы этого избежать каждый участник должен иметь собственный аудиопоток, который не содержит его собственного звука. Сначала аудиокодеры декодируют полезную нагрузку RTP в аудиообразцы. Затем аудиомикшер смешивает аудиообразцы в отдельные потоки аудиообразцов, как показано на Рисунок 10. Потом отдельный аудиокодер кодирует аудиообразцы обратно в полезную нагрузку RTP [19].

7.3. Сравнение архитектур

Таблица 1 – Сравнение Mesh, MCU, SFU архитектур

	Mesh	MCU	SFU
in/out streams	3/3	1/1	3/1
input traffic	3 Мбит/с	1 Мбит/с	3 Мбит/с
output traffic	3 Мбит/с	1 Мбит/с	1 Мбит/с
client CPU	3 encoder + 3 decoder = 60%	1 encoder + 1 decoder = 20%	1 encoder + 3 decoder = 40%
server CPU	0	100%	10%
latency	min	max	avg
SIP/live	–	+	–
max participants	8	∞	50

В Таблице 1 представлено сравнение Mesh, MCU, SFU архитектур по различным параметрам.

Mesh

Mesh-топология выгодна за счет прямого соединения участников без вмешательства сервера, что сокращает издержки.

Однако с ростом числа участников квадратично растет входящий и исходящий трафик. Создается большая нагрузка на CPU клиентского устройства, потому что большинство устройств поддерживают аппаратно ускоренное кодирование только одного потока в один момент, а тут надо кодировать и декодировать сразу несколько потоков.

MCU

Основное преимущество MCU – число участников ограничено только количеством потоков, которое можно смикшировать на сервере. Кроме того, MCU экономит ресурсы на клиенте и позволяет реализовать трансляцию или запись на сервере.

Однако в MCU требуется много ресурсов на сервере, а также задержка в этой архитектуре самая большая.

SFU

SFU архитектура выгодна за счет того, что часть нагрузки переносится с сервера на клиент. Кроме того исходящий трафик постоянен, а входящий растет пропорционально числу участников видеоконференции.

Главный минус SFU – это ограниченное число максимальных участников, которое, конечно, выше, чем в Mesh архитектуре, но меньше, чем в MCU. Это происходит из-за дорогого микширования входящих потоков, которое ложится на клиентов [20].

ЗАКЛЮЧЕНИЕ

В рамках этой научно-исследовательской работы мы рассмотрели историю развития видеоконференций, начиная с их зарождения в 1920-ых до современных технологий. Подробно рассмотрели протокол WebRTC и его возможности для обеспечения качественной коммуникации между участниками видеоконференций. Определили ожидания конечных пользователей и проблемы, влияющие на качество видеоконференций, а также рассмотрели, как WebRTC справляется с задачей передачи видео.

Одной из ключевых тем исследования было масштабирование участников видеоконференций. В частности, мы рассмотрели проблемы, связанные с ограничениями P2P архитектуры WebRTC и альтернативы, позволяющие решить проблему с масштабированием.

В будущих исследованиях мы подробно рассмотрим MCU (Multiple Control Unit) архитектуру для WebRTC, а также сосредоточимся на разработке и оптимизации приложения, устойчивого к условиям постоянного увеличения числа участников в видеоконференции.

Список использованных источников

[1] https://www.techtarget.com/whatis/feature/The-history-and-evolution-of-video-conferencing.

[2] Congestion Control for WebRTC: Standardization Status and Open Issues // IEEE, 2017.

[3] https://webrtcforthecurious.com/docs/01-what-why-and-how/.

[4] https://habr.com/ru/companies/odnoklassniki/articles/479852.

[5] https://bloggeek.me/webrtcglossary/bwe/.

[6] https://bloggeek.me/webrtcglossary/remb/.

[7] https://bloggeek.me/webrtcglossary/transport-cc/.

[8] https://en.wikipedia.org/wiki/Packet_loss_concealment.

[9] https://bloggeek.me/webrtcglossary/packet-loss/.

[10] https://bloggeek.me/webrtcglossary/fec/.

[11] https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D1%81%D0%B5%D1%82%D0%B8.

[12] https://bloggeek.me/webrtcglossary/jitter/.

[13] https://webrtcforthecurious.com/docs/05-real-time-networking/.

[14] https://bloggeek.me/webrtcglossary/mtu-size/.

[15] Packet Reordering Correction for Low-Latency Network Applications // IEEE, 2022.

[16] https://webrtcforthecurious.com/docs/06-media-communication/.

[17] https://www.digitalsamba.com/blog/p2p-sfu-and-mcu-webrtc-architect

[18] A scalable WebRTC-based framework for remote video collaboration applications, 2018.

[19] A P2P-MCU Approach to Multi-Party Video Conference with WebRTC, 2014.

[20] https://habr.com/ru/companies/vk/articles/575358/.

[21] https://webrtcforthecurious.com/docs/02-signaling/.

[22] https://webrtcforthecurious.com/docs/03-connecting/.

[23] https://webrtcforthecurious.com/docs/04-securing/.

[24] https://webrtcforthecurious.com/docs/07-data-communication/.