Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 20Б.11-мм

Шлегель Алина Владимировна

Исследование способов беспроводной потоковой передачи видео между Android-устройствами с минимальной задержкой

Отчёт по учебной практике

в форме «Сравнение»

Научный руководитель:
старший преподаватель каф. СП Я.А. Кириленко

Консультанты:
инженер-исследователь Сколтех А.В. Корнилова
Программист-разработчик VK Д.С. Ярош

Санкт-Петербург
2022

Оглавление

1 Введение

В современном мире появляется всё больше приложений и алгоритмов, основанных на совместном использовании нескольких смартфонов одновременно — особенно явно это ощущается в нише, связанной с компьютерным зрением, где смартфоны используются для съемки со снятием данных с их датчиков. Такая востребованность вызвана сразу несколькими факторами. Во-первых, в современных смартфонах активно развиваются технологии съемки — по сравнительно низкой цене пользователям становится доступно достаточно высокое качество получаемых снимков. Во-вторых, в смартфоны внедряются новейшие технологий передачи данных такие как, например, Wi-Fi с частотой передачи 5 Гигагерц. В-третьих, производительность мобильных устройств постоянно растет, что позволяет решать с их помощью всё более вычислительно сложные проблемы.

Уже сейчас совместное использование нескольких смартфонов используется при решении таких задач как Multi-User AR [6, 1], где устройства координируются между собой для корректной демонстрации общих виртуальных объектов. Также оно применяется в областях, где используются карты глубины, создаваемые из стерео-изображений [19, 9]: 3D-реконструкция сцены, оцифровка объектов реального мира и многие другие — в них требуется одновременно получать снимки с нескольких позиций в пространстве.

В основе совместной работы смартфонов лежит передача данных между ними. Частым требованием здесь является отсутствие проводного соединения между устройствами, так как это значительно повышает их мобильность. Также зачастую необходимо проводить передачу в мягком реальном времени — особенно данное требование проявляется в задачах, связанных со съемкой, так как в противном случае конечный пользователь может легко заметить рассинхронизированность изображений, а при последующей обработке полученных с задержкой данных могут проявиться неточности из-за их несоответствия. Проблема становится всё более заметной при росте объема передаваемых данных,

а разделить их на части при этом возможно далеко не всегда. К примеру, подсчет карты глубины невозможен без передачи изображения целиком на одно устройство, которое и будет проводить вычисления. Таким образом, решение задачи беспроводной потоковой передачи видео с минимальной задержкой является одним из высоко востребованных в современных реалиях.

В данной работе проводится исследование существующих технологий, позволяющих осуществлять беспроводную потоковую передачу видео в мягком реальном времени между устройствами под управлением системы Android. При помощи данного исследования появляется возможность найти решение проблем, возникающих при совместной работе Android-смартфонов над задачами, связанными со съемкой и требующими для своего решения минимизации задержки между передаваемыми кадрами.

2 Постановка задачи

Целью данной работы является обзор и сравнение технологий беспроводной передачи видеоконтента в режиме мягкого реального времени. Для её выполнения были поставлены следующие задачи:

- выполнить обзор протоколов передачи видеоконтента в режиме реального времени и подходов к их реализации;
- выполнить обзор существующих реализаций найденных методов;
- оптимизировать выбранные реализации;
- провести обзор принятых в данной области метрик и сравнение выбранных реализаций с их использованием.

3 Обзор

В данной секции рассмотрены способы беспроводной передачи видеоконтента в режиме мягкого реального времени, а также их реализации для операционной системы Android.

3.1 Технология беспроводной передачи данных

Для беспроводной передачи видео могут быть использованы такие технологии как Bluetooth и Wi-Fi. В современных смартфонах, как правило, имеются модули, обеспечивающие работу обеих названных технологий, однако для осуществление передачи видео в реальном времени Bluetooth не предназначен из-за его низкой скорости передачи, поэтому в данной работе рассматривается технология Wi-Fi — её скоростные показатели достаточны для поставленной задачи.

3.2 Протоколы потоковой передачи данных

Для потоковой передачи видео в беспроводной сети используются протоколы транспортного уровня для доставки данных и протоколы прикладного уровня для передачи служебной информации. Схема взаимосвязи протоколов приведена на Рис. 1. Здесь и далее используется сетевая модель OSI[1], в качестве протокола сетевого уровня используется IP[2] как повсеместно используемый для данных задач в современном мире.

3.2.1 Транспортный уровень

Для передачи данных по сети используют следующие транспортные протоколы:

- TCP [12] — протокол, гарантирующий целостность передаваемых данных за счет предварительной установки соединения трех-этапным

[1] https://en.wikipedia.org/wiki/OSI_model
[2] https://en.wikipedia.org/wiki/Internet_Protocol

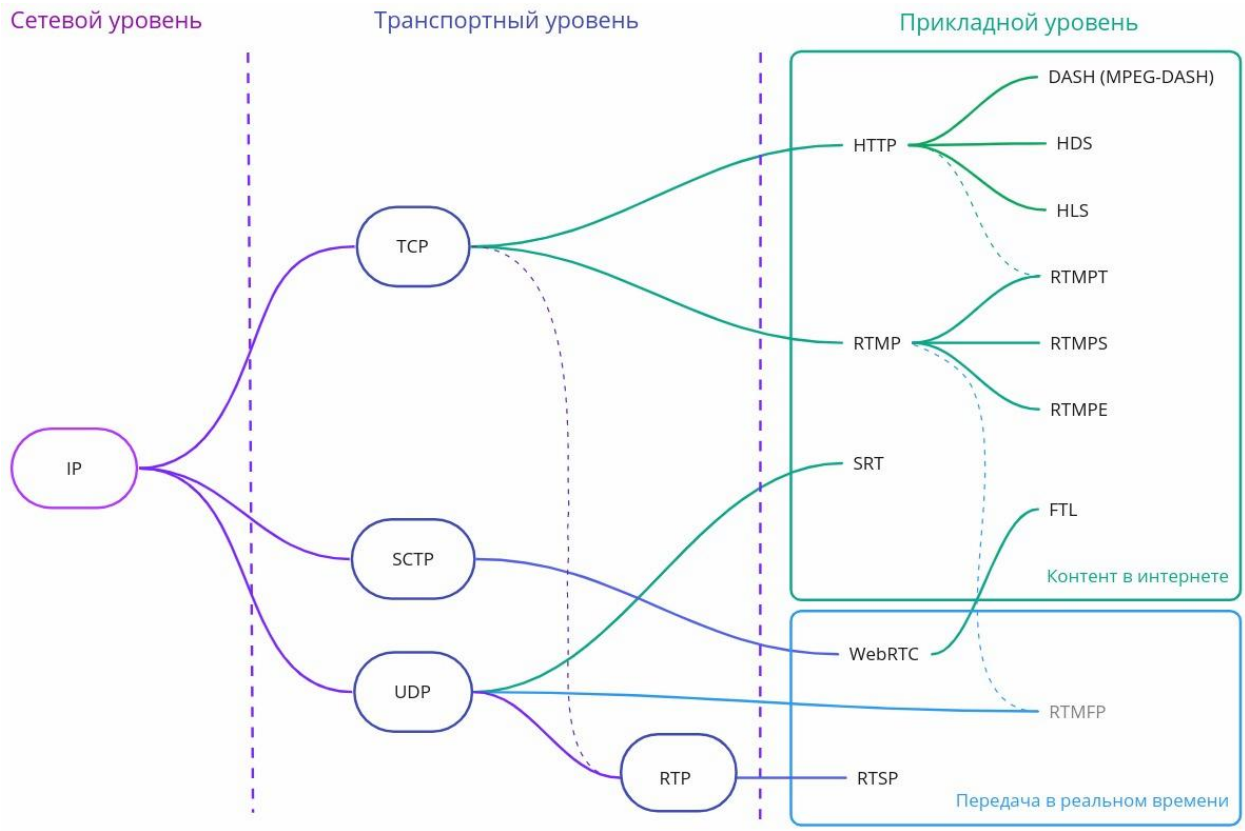

Рис. 1: Популярные протоколы для передачи видеоконтента.

рукопожатием (three-way handshake) и осуществления повторных запросов в случае потери или удаления полученных дублированных пакетов;

- UDP [13] — протокол, не осуществляющий контроль за целостностью доставляемых данных, за счет чего при его использовании можно достичь прироста по скорости их передачи;

- SCTP [10] — протокол повышенной защищенности с механизмами четырех-этапного рукопожатия (four-way handshake) и ввода маркера (cookie).

Также зачастую в качестве транспортного протокола выделяется RTP [5]. Он является надстройкой над транспортными протоколами, адаптирующей их к потоковой передаче, при этом сам по себе RTP не способен устанавливать или прерывать соединения. Несмотря на то, что согласно спецификации, RTP основан на TCP, как правило, его реализации базируются на UDP. Это обусловлено тем, что надежность

передачи данных, гарантируемая TCP, влечёт дополнительные временные задержки. Особенностью RTP является наличие в отправляемых с помощью него сообщениях заголовков с временными метками, которые могут быть использованы для синхронизации, а также номеров пакета, что применяется для восстановления порядка их отправки. Всё это активно применяется при передаче видеоконтента.

Таким образом, для потоковой передачи видео в качестве транспортного протокола наилучшим образом подходит RTP.

3.2.2 Прикладной уровень

Задачи по передаче видеоконтента, решаемые протоколами прикладного уровня, можно разделить на две группы: передача видео в реальном времени и его воспроизведение интернете.

Протоколы передачи видео в реальном времени преимущественно используются для видеозвонков и видеонаблюдения. В этих задачах требуется низкая задержка — как правило для применяемых при их решении протоколов, характерна задержка менее одной секунды.

Для видеозвонков наибольшее распространение получили следующие протоколы.

- WebRTC [20] — проект, осуществляющий аудио- и видеосвязь внутри веб-страниц. Данные передаются с использованием SCTP, что увеличивает временную задержку из-за предоставления безопасности соединения.

- RTMFP [11] — первоначально проприетарный протокол, разработанный Adobe Systems и опубликовали в 2014 году.

На текущий момент WebRTC заменил RTMFP в большинстве крупных приложений, предоставляющих возможность использования видеочата, таких как, например, Skype, Zoom и Telegram. При разработке новых систем, как правило, предпочтение также отдают WebRTC.

Для видеонаблюдения посредством IP-камер традиционно используют RTSP [8], основанный на RTP/UDP. RTSP расширяет возмож-

ности RTP засчет добавления в сообщения дополнительной служебной информации, позволяя тем самым лучше контролировать соединение.

Протоколы для воспроизведения видео в интернете используются для транслирования видеоконтента с удаленного сервера в браузер конечного пользователя. К данной группе относятся следующие протоколы:

- HTTP [23] — текстовый протокол, использующий TCP на транспортном уровне. Данный протокол не хранит состояние и не предоставляет информацию о предыдущих запросах. На HTTP основаны такие протоколы как HLS [7], DASH (MPEG-DASH) [22] и HDS [21].

- FTL [24] — протокол, основанный на WebRTC. Разработан специально для платформы Mix[3].

- RTMP [25] — протокол, основанный на TCP. Он поддерживает постоянные соединения, передавая сообщения с динамически изменяемым размером. К вариациям RTMP относятся: RTMPT (тунелирует RTMP через HTTP), RTMPS (зашифрован через SSL), RTMPE (зашифрован с помощью механизма безопасности Adobe) и RTMFP (использует UDP вместо TCP).

- SRT [26] — обеспечивает соединение и управление, а также надежную передачу, аналогичную TCP, на прикладном уровне, используя при этом UDP на транспортном уровне.

Для задачи воспроизведения видео в интернете не так важна задержка — акцент делается на безопасность и надежность соединения, а также на качество получаемого пользователем контента. Вследствие этого данные протоколы преимущественно используют TCP на транспортном уровне из-за его надежности по сравнению с UDP. SRT, основанный на UDP, также реализует дополнительные рукопожатия, но не на транспортном уровне, а на прикладном, что негативно сказывается на задержке. По данным видеоплатформы для бизнес-приложений

[3] https://mix.com

Wowza[4] задержка, присущая рассмотренным протоколам, находится в диапазоне от 1 секунды до 45 секунд.

3.2.3 Выводы о выборе протокола

Из рассмотренных протоколов потоковой передачи данных для транслирования видео с минимальной задержкой наилучшим образом подходит RTSP. Во-первых, он может быть реализован поверх UDP, что позволяет сократить время передачи контента за счет снижения уровня контроля за целостностью данных. А во-вторых, он основан на RTP, адаптированном для потоковой передачи. Всё это делает его наиболее быстрым и удобным решением в рамках поставленной задачи.

3.3 RTSP

Real Time Streaming Protocol — протокол контроля передачи данных в режиме реального времени. Он объединяет в себе передачу данных и сигналов управления. На устройстве RTSP функционирует на двух разных портах: через первый передается видео-поток (посредством RTP), а через второй — служебная информация, используемая для управления передачей. Сигналы, поступающие через второй порт, переключают состояния медиа-сервера[5] и клиента[6], что позволяет устанавливать параметры передачи и продолжать управлять медиапотоком в течение длительного времени (до следующего переключения). Важной особенностью RTSP является тот факт, что запросы может отправлять как клиент, так и медиа-сервер.

Чтобы избежать путаницы в названиях, здесь и далее **Сервером** будем называть источник передаваемого видео, а **Плеером** — получателя, который будет воспроизводить полученное видео.

[4] https://www.wowza.com/wp-content/uploads/latency-continuum-2021-with-protocols-700x300-1.png

[5] часть устройства протокола, предоставляющая данные.

[6] часть устройства протокола, запрашивающая данные.

4 Исследование существующих реализаций

В данном разделе рассмотрены RTSP Серверы и Плееры, реализованные для платформы Android.

4.1 RTSP Серверы

Были рассмотрены следующие реализации RTSP Серверов для Android.

Rtmp-rtsp-stream-client-java [34] — самая популярная из рассмотренных библиотека с активной поддержкой со стороны авторов и сообщества. В ней помимо прочего реализован RTSP поверх UDP, при этом осуществляется поддержка нескольких аппаратных кодировщиков видео[7], а именно H.264 и H.265.

Libstreaming [30] — данная библиотека специализируется исключительно на RTSP/UDP. Она реализует кодирование через H.263 и H.264. В ней отсутствует поддержка Camera2 API[8], что существенно ограничивает её использование совместно с камерой на современных Android-устройствах.

Sms [35] — реализация, поддерживающая сразу несколько протоколов, основанных на TCP, в том числе и RTSP. В качестве аппаратного кодировщика используется H264.

Из рассмотрения были исключены следующие Серверы:

RTSP-Camera-for-Android [36] — проект, написанный в 2012 году и с тех пор не обновлявшийся.

Live264Streamer [31] — опубликованную версию данного проекта не удалось запустить ни на одном из имеющихся Android-устройств[9];

LiveStreamer [33] — неподходящая функциональность: передает трансляцию изображения экрана устройства;

RTSPMultiCam [32] — неподходящая функциональность: передает комбинированное изображение с нескольких веб-камер;

[7] https://en.wikipedia.org/wiki/Video_codec

[8] https://developer.android.com/training/camera2

[9] Использованные здесь и далее Android-устройства подробно описаны в секции, посвященной экспериментам

RTSP.Server.Android [17] — проект, поддержка которого прекращена более 6 лет назад; помимо этого, его опубликованную версию не удалось запустить ни на одном из имеющихся Android-устройств.

Основное требование к Серверам — использование RTSP через транспортный протокол RTP/UDP. Также реализация должна поддерживать современный Android API. Данное требование важно, в частности, для возможности использования Camera2 API, предоставляющего более богатый набор возможностей по взаимодейтсвию с камерами устройств. Также новые версии лучше оптимизированы, в них реализовано больше возможностей для работы с видеоконтентом таких как, например, MediaCodec[10].

Из найденных Серверов, только rtmp-rtsp-stream-client-java соответствует всем указанным требованиям.

4.2 RTSP Плееры

В отличие от RTSP Серверов, реализации RTSP Плееров преимущественно коммерческие [16, 4] — это существенно ограничивает возможности по их интеграции в сторонние проекты. Реализаций с открытым исходным кодом же сравнительно мало, среди них было найдено лишь две полнофункциональные и поддерживаемые на момент проведения исследования: LibVLC [18] и Ffmpeg [3].

4.2.1 LibVLC

LibVLC состоит из множества подключаемых частей (плагинов), что позволяет разработчикам создавать широкий спектр мультимедийных приложений, используя возможности медиаплатформы VLC [15]. Данная библиотека позволяет ограничивать кеширование данных сети и протокола и предоставляет несколько опций контроля синхронизации и качества видео-передачи. Контроль происходит посредством следующих параметров[11]:

[10] https://developer.android.com/reference/android/media/MediaCodec

[11] Для рассмотрения были выбраны параметры, указанные в документации VLC и предположительно оказывающие влияние на задержку при передаче видео-потока

- `network-caching` — значение кеширования для сетевых ресурсов в миллисекундах;
- `rtsp-caching` — дополнительное описание в документации отсутствует;
- `clock-jitter` — определяет максимальное фазовое дрожание цифрового сигнала входных данных (jitter), которое алгоритмы синхронизации компенсируют;
- `clock-synchro` — возможность синхронизации входных часов для источников в реальном времени, позволяющая избежать рывков при воспроизведении сетевых потоков.

В рамках данной работы рассматривалась реализация vlc-android [14], написанная на основе LibVLC для платформы Android и обладающая большой функциональностью. Для взаимодействия с ней был реализован Сервер на основе LibVLC, специализирующийся исключительно на воспроизведении видео-потока, что делает его более гибким по сравнению со стандартным плеером VLC.

4.2.2 Ffmpeg

Ffmpeg — это мультимедийная платформа для записи, преобразования и потоковой передачи аудио и видео. Важной частью Ffmpeg является Ffplay — консольный инструмент, используемый для воспроизведения потокового видео и изначально разработанный для тестового стенда Ffmpeg. Ffplay обладает обширным набором опций[12], позволяющих управлять передачей — в частности, с их помощью можно уменьшить задержку передачи видео-потока.

- `fflags nobuffer` — уменьшает задержку, вызванную буферизацией при анализе начальных входных потоков;

[12]Для рассмотрения были выбраны параметры, указанные в документации Ffmpeg и предположительно оказывающие влияние на задержку при передаче видео-потока

- `analyzeduration` — задает количество микросекунд, затрачиваемое на проверку входного сигнала;
- `flush_packets` — очищает базовый поток ввода-вывода после каждого пакета (при включении позволяет уменьшить задержку);
- `probesize` — задает размер данных, который необходимо проанализировать для получения информации о потоке (проведение данного анализа может увеличить задержку);
- `fast` — включает дополнительные оптимизации, не соответствующие спецификации реализации;
- `low_delay` — принудительно уменьшает задержку;
- `reorder_queue_size` — устанавливает количество пакетов в буфере для обработки переупорядоченных пакетов RTSP.

Для Android существует единственная реализация Ffplay — **ijkplayer** [28]. Данный проект переносит на Android все библиотеки Ffmpeg, необходимые для получения и воспроизведения потокового видео, а также позволяет встраивать в себя произвольный код. На ее основе были созданы следующие реализации.

Yuneec-RTSP-Player-Android [27] — библиотека, созданная, чтобы упростить взаимодействие с ijkplayer. На данный момент проект более не поддерживается, на имеющихся Android-устройствах предоставляемый в нём Плеер запустить не удалось.

GSYVideoPlayer [2] — развивающаяся библиотека с реализацией Плеера. На момент проведения исследования рабочая реализация не предоставлена.

Rtsp_player [29] — Плеер, разработанный в 2017 году и более не обновляемый. На имеющихся Android-устройствах данный Плеер запустить не удалось.

Помимо указанных проектов авторы библиотеки ijkplayer самостоятельно реализовали пример Плеера, однако он не обновлялся с 2017

года, а запустить его на имеющихся Android-устройствах не удалось. Сама библиотека ijkplayer документации по работе с ней не содержит.

Таким образом, работоспособных реализаций Плеера, основанных на библиотеке Ffmpeg, для системы Android найдено не было.

5 Методология проведения измерений

В данном разделе описана методология измерения задержки передачи видеоконтента для различных конфигураций Плееров и Сервера. Она призвана повысить точность получаемых замеров и обеспечить их проведения в наиболее схожих условиях.

5.1 Проведение замеров

Для измерения задержки передачи видео требуется два Android-смартфона с установленным RTSP Сервером на одном из них и Плеером — на другом, а также цифровые часы и устройство для съемки экранов смартфонов. Установка для измерения выглядит следующим образом (она изображена на Рис. 2): смартфон-сервер проводит съемку часов и транслирует данное видео на смартфон-плеер, в это время экраны обоих смартфонов (на них демонстрируются захватываемое и получаемое видео соответственно) снимаются на видео другим устройством. При этом передаваемое видео воспроизводится с некоторой задержкой, что и запечатлевается снимающим устройством. Параметры видео (такие как частота изменения кадров и разрешение) для всех замеров используются одни и те же, что обеспечивает однородность результатов.

5.2 Обработка результатов

Для обработки полученной видеозаписи экранов двух смартфонов необходимо выбрать кадры с равными промежутками, чтобы оценить задержку и вывести её зависимость от времени. Такая зависимость демонстрирует влияние тестируемых параметров на задержку передачи видео-потока.

Для проведения обработки был написан Python-скрипт, который принимает на вход полученную видеозапись и возвращает выбранный набор кадров.

Рис. 2: Установка для проведения замеров наблюдаемой задержки передачи видео-потока.

6 Измерение задержки передачи видео

В данном разделе описаны эксперименты по определению влияния различных конфигурационных параметров на задержку передачи видео-потока. Эксперименты проводились по методологии, описанной в соответствующем разделе.

6.1 Тестирующая установка

Для замеров были использованы следующие устройства:

- смартфон с Сервером, основанным на библиотеке rtmp-rtsp-stream-client-java (запись проводилась с кадровой частотой 30 кадров в секунду) — Android Samsung Galaxy S10e;
- смартфон с Плеером, основанным на LibVLC — Android Samsung Galaxy S10 Lite;
- монитор (частота обновления — 60 Герц) с часами, замеряющими

сантисекунды[13];

- записывающее устройство (запись проводилась с кадровой частотой 30 кадров в секунду) — Samsung Galaxy Tab S6 Lite.

Между смартфонами, участвующими в эксперименте, видео передавалось с частотой 30 кадров в секунду и разрешением 720 на 480 пикселей.

6.2 Погрешность тестирующей установки

Для дальнейшего исследования необходимо оценить погрешность измеряющей установки в целом.

- Монитор с часами с сантисекундами с кадровой частотой 60 кадров в секунду может отставать от реального времени на *две* сантисекунды.

- Видео со смартфона с Сервером, снимающего монитор, может отставать на *шесть* сантисекунд из-за задержки монитора и кадровой частоты записи 30 кадров в секунду.

- Смартфон с Плеером получает видео с монитора с задержкой *две* сантисекунды и демонстрирует его с кадровой частотой 30 кадров в секунду, что дает суммарную задержку до *шести* сантисекунд.

- Записывающее устройство снимает видео с кадровой частотой 30 кадров в секунду, на данном видео каждый из смартфонов показывает время с задержкой до 6 сантисекунд.

Итого, общая погрешность измерений, полученных при помощи описанной установки, равная разности двух показателей времени со смартфонов с Сервером и Плеером, не будет превосходить ± 6 сантисекунд.

[13] https://htmlblog.ucoz.net/html_files/time.html

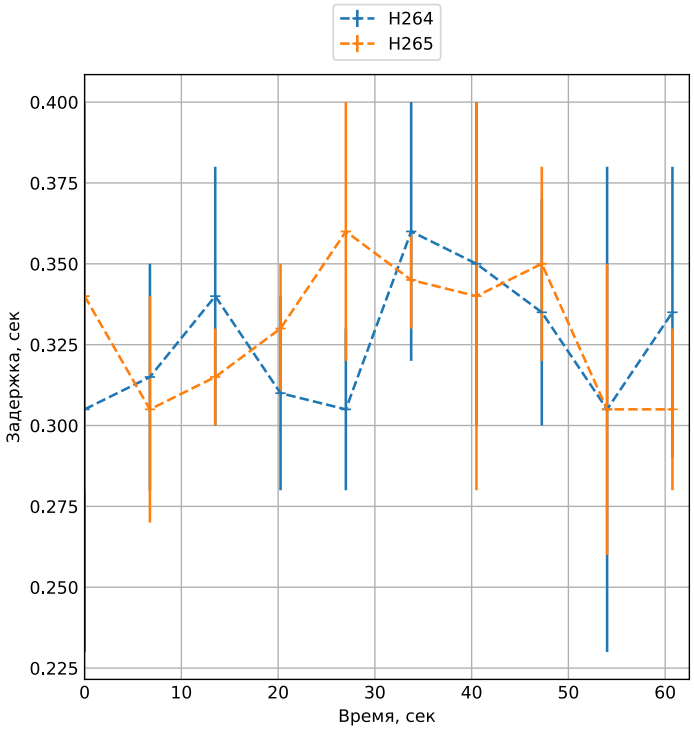

Рис. 3: График зависимости задержки видео-потока от выбранного на Сервере аппаратного кодировщика видео.

6.3 Результаты

В данной секции описаны результаты проведенных экспериментов.

Для каждого исследуемого параметра проводилось по шесть видеозаписей длительностью около минуты. На графиках, продемонстрированных в данной секции, показаны средние, минимальные и максимальные значения задержки, полученной в ходе соответствующего замера.

6.3.1 Влияние параметров Сервера

У Сервера присутствует всего один изменяемый конфигурационный параметр — это аппаратный кодировщик видео с доступными значениями H.264 и H.265. Как видно из графика, изображенного на Рис. 3, данный параметр не влияет на задержку при передаче видео-потока.

6.3.2 Влияние параметров Плеера

Плеер обладает множеством конфигурационных параметров, для каждого из которых были в ходе эксперимента были найдены зави-

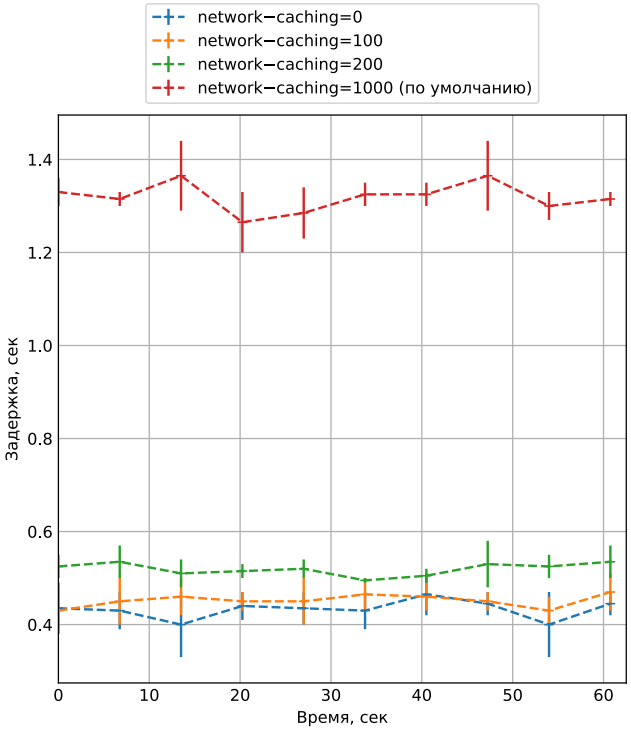

Рис. 4: График зависимости задержки видео-потока от значения параметра Плеера network-caching.

симости величины задержки передачи видео-потока от его изменения.

`network−caching` — на Рис. 4 приведены результаты замеров задержки при выставлении различных значений данного параметра. По умолчанию его значение равняется 1000, что сильно сказывается на задержке: чем меньше его значение, тем меньше наблюдаемое отставание. Однако при приближении значения к нулю задержка практически перестает изменяться, что видно на Рис. 5.

Все последующие замеры проведены с установленным параметром `network−caching`=0, так как данное значение в среднем дало самые низкие показания задержки.

`rtsp−caching` — наблюдаемая задержка прямо зависит от значения данного параметра. Наилучший результат показала конфигурация со значением 100, что продемонстрировано на Рис. 6. В последующих замерах был выставлен параметр `rtsp−caching`=100.

`clock−jitter` — наблюдаемая задержка прямо зависит от значения данного параметра, что продемонстрировано на Рис. 7. Самый лучший результат был получен при значении данного параметра, рав-

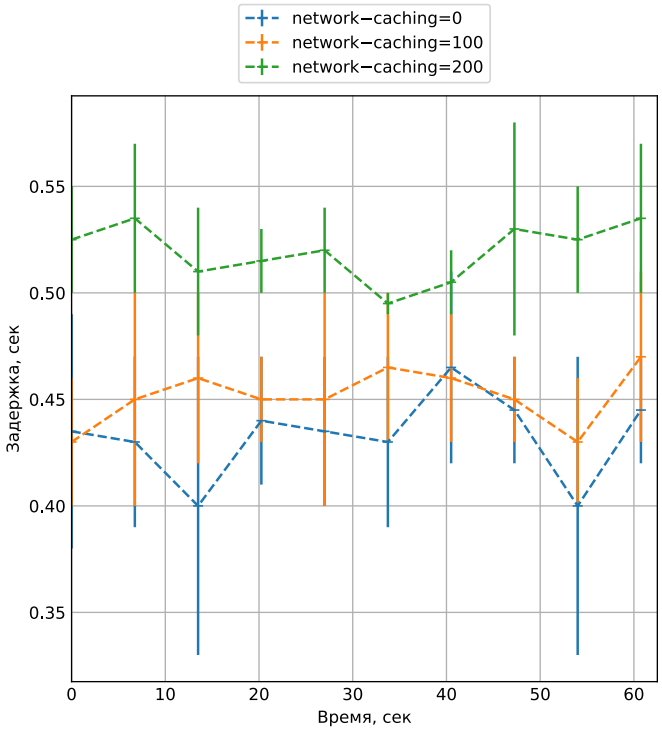

Рис. 5: График зависимости задержки видео-потока от значения параметра Плеера network-caching со значениями близкими к 0.

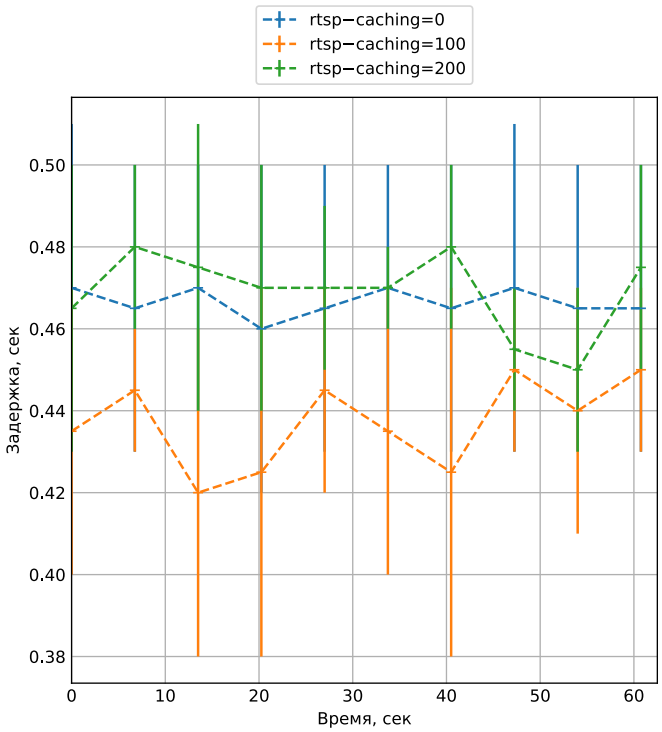

Рис. 6: График зависимости задержки видео-потока от значения параметра Плеера rtsp-caching.

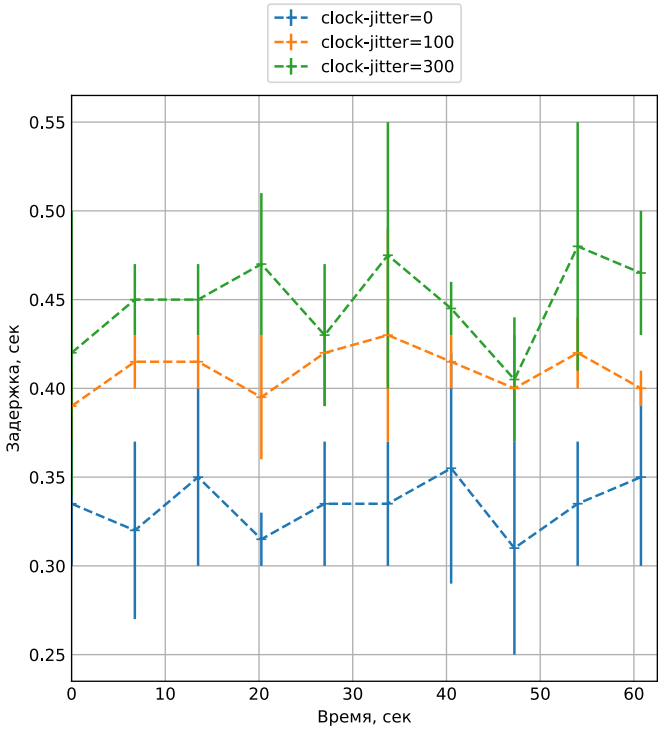

Рис. 7: График зависимости задержки видео-потока от значения параметра Плеера clock-jitter.

ном нолю, поэтому в последующих замерах был выставлен параметр `clock-jitter`=0.

`clock-synchro` — включение данного параметра положительно сказалось на величине задержки, что продемонстрировано на Рис 8.

6.3.3 Задержка при выставлении оптимальных значений параметров

При выставлении всех приведенных выше параметров Сервера и Плеера в значения, при которых была получена наименьшая задержка, на использованных смартфонах удалось добиться задержки менее 0.4 секунд. Причем, данное значение не увеличивается с течением времени. Данный результат продемонстрирован на Рис. 9.

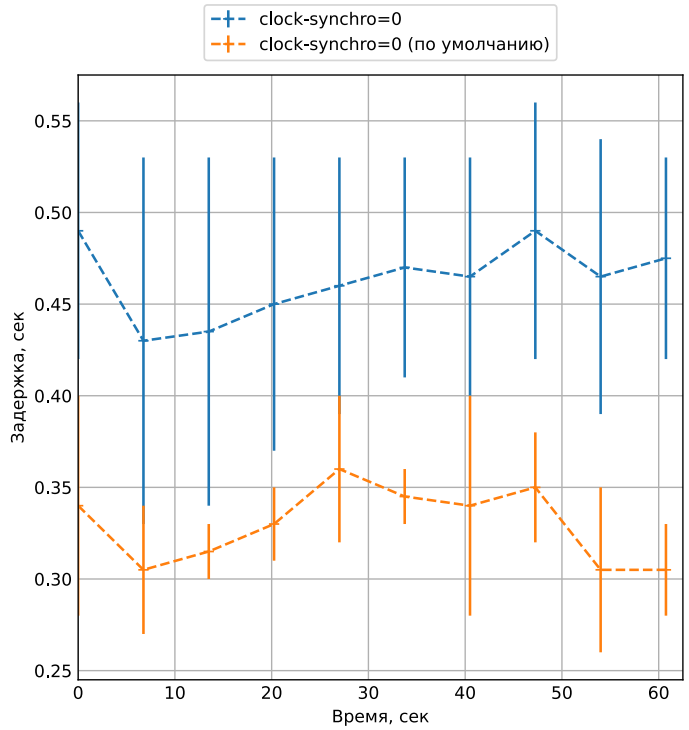

Рис. 8: График зависимости задержки видео-потока от значения параметра Плеера clock-synchro.

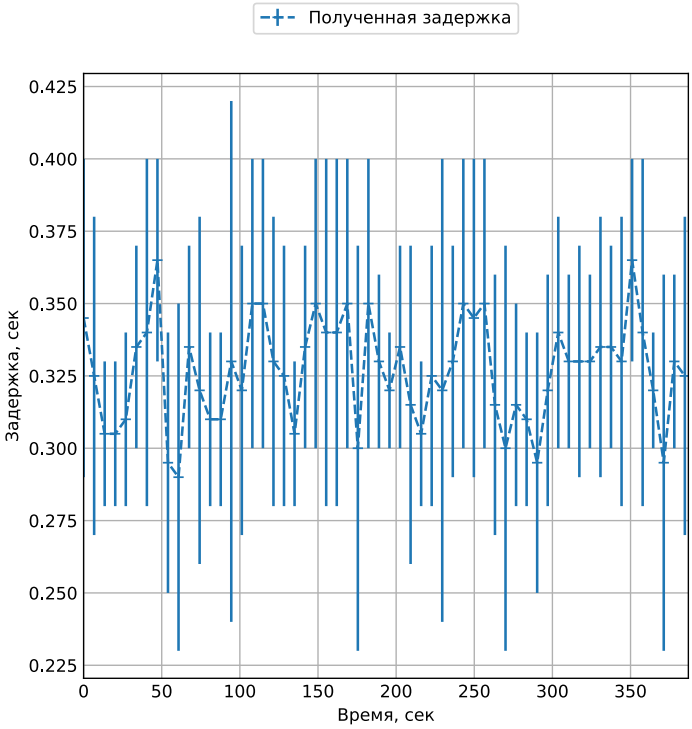

Рис. 9: График зависимости задержки видео-потока от времени при выставлении всех рассмотренных параметров в оптимальные значения.

7 Заключение

В результате работы были получены следующие результаты:

- выполнен обзор протоколов передачи видеоконтента в режиме реального времени и подходов к их реализации;
- выполнен обзор существующих реализаций найденных методов;
- проведена оптимизация выбранных реализаций;
- проведены обзор принятых в данной области метрик и сравнение выбранных реализаций с их использованием.

Результаты работы расположены в данном репозитории.

Список литературы

[1] Can 5G mmWave Support Multi-user AR? / Moinak Ghoshal, Pranab Dash, Zhaoning Kong et al. // Passive and Active Measurement / Ed. by Oliver Hohlfeld, Giovane Moura, Cristel Pelsser. — Cham : Springer International Publishing, 2022. — P. 180–196.

[2] CarGuo. GSYVideoPlayer: Video player. — https://github.com/CarGuo/GSYVideoPlayer. — [Online; accessed 19-May-2022].

[3] Ffmpeg. — [Online; accessed 19-May-2022]. URL: https://ffmpeg.org/.

[4] IP Webcam Pro - Apps on Google Play. — https://play.google.com/store/apps/details?id=com.pas.webcam.pro&hl=en&gl=US. — (Accessed on 05/21/2022).

[5] Lazzaro John. Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport. — RFC 4571. — 2006. — jul. — URL: https://www.rfc-editor.org/info/rfc4571.

[6] Multi-User Augmented Reality with Communication Efficient and Spatially Consistent Virtual Objects / Xukan Ran, Carter Slocum, Yi-Zhen Tsai et al. // Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies. — New York, NY, USA : Association for Computing Machinery, 2020. — P. 386–398. — ISBN: 9781450379489. — URL: https://doi.org/10.1145/3386367.3431312.

[7] Pantos Roger, May William. HTTP Live Streaming. — RFC 8216. — 2017. — aug. — URL: https://www.rfc-editor.org/info/rfc8216.

[8] Rao Anup, Lanphier Rob, Schulzrinne Henning. Real Time Streaming Protocol (RTSP). — RFC 2326. — 1998. — apr. — URL: https://www.rfc-editor.org/info/rfc2326.

[9] Shamsafar Faranak, Zell Andreas. TriStereoNet: A Trinocular Framework for Multi-baseline Disparity Estimation. — 2021. — URL: https://arxiv.org/abs/2111.12502.

[10] Stewart Randall R. Stream Control Transmission Protocol. — RFC 4960. — 2007. — sep. — URL: https://www.rfc-editor.org/info/rfc4960.

[11] Thornburgh Michael C. Adobe's RTMFP Profile for Flash Communication. — RFC 7425. — 2014. — dec. — URL: https://www.rfc-editor.org/info/rfc7425.

[12] Transmission Control Protocol. — RFC 793. — 1981. — sep. — URL: https://www.rfc-editor.org/info/rfc793.

[13] User Datagram Protocol. — RFC 768. — 1980. — aug. — URL: https://www.rfc-editor.org/info/rfc768.

[14] VLC for Android. — https://github.com/videolan/vlc-android. — (Accessed on 05/23/2022).

[15] VLC media player. — https://www.videolan.org/vlc/index.html. — (Accessed on 05/23/2022).

[16] VXG: IP Camera Viewer App – Apps on Google Play. — https://play.google.com/store/apps/details?id=org.rtspplr.app. — (Accessed on 05/21/2022).

[17] VideoExpertsGroup. RTSP.Server.Android. — https://github.com/VideoExpertsGroup/RTSP.Server.Android. — [Online; accessed 19-May-2022].

[18] VideoLAN. LIBVLC. — [Online; accessed 19-May-2022]. URL: https://www.videolan.org/vlc/libvlc.html.

[19] Wang Rui, Schwörer Martin, Cremers Daniel. Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras // 2017

IEEE International Conference on Computer Vision (ICCV). — 2017. — P. 3923–3931.

[20] WebRTC 1.0: Real-time communication between browsers. — [Online; accessed 19-May-2022]. URL: https://www.w3.org/TR/webrtc/.

[21] Wikipedia contributors. Adaptive bitrate streaming — Wikipedia, The Free Encyclopedia. — https://en.wikipedia.org/wiki/Adaptive_bitrate_streaming. — 2022. — [Online; accessed 19-May-2022].

[22] Wikipedia contributors. Dynamic Adaptive Streaming over HTTP — Wikipedia, The Free Encyclopedia. — https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP. — 2022. — [Online; accessed 19-May-2022].

[23] Wikipedia contributors. Hypertext Transfer Protocol — Wikipedia, The Free Encyclopedia. — https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. — 2022. — [Online; accessed 19-May-2022].

[24] Wikipedia contributors. Mixer (service) — Wikipedia, The Free Encyclopedia. — https://en.wikipedia.org/wiki/Mixer_(service). — 2022. — [Online; accessed 19-May-2022].

[25] Wikipedia contributors. Real-Time Messaging Protocol — Wikipedia, The Free Encyclopedia. — https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol. — 2022. — [Online; accessed 19-May-2022].

[26] Wikipedia contributors. Secure Reliable Transport — Wikipedia, The Free Encyclopedia. — https://en.wikipedia.org/wiki/Secure_Reliable_Transport. — 2022. — [Online; accessed 19-May-2022].

[27] YUNEEC. Yuneec-RTSP-Player-Android: Android Library for displaying live video stream from camera. — https://github.com/YUNEEC/Yuneec-RTSP-Player-Android. — [Online; accessed 19-May-2022].

[28] bilibili. ijkplayer: Android/iOS video player based on FFmpeg n3.4, with MediaCodec, VideoToolbox support. — https://github.com/Bilibili/ijkplayer. — [Online; accessed 19-May-2022].

[29] bowen919446264. rtsp_player: play rtsp stream by use ijkplayer. — https://github.com/bowen919446264/rtsp_player. — [Online; accessed 19-May-2022].

[30] fyhertz. libstreaming: A solution for streaming H.264, H.263, AMR, AAC using RTP on Android. — https://github.com/fyhertz/libstreaming/. — [Online; accessed 19-May-2022].

[31] huzongyao. Live264Streamer: A Project To Study RTSP Streamer H264 Video With Live555. — https://github.com/huzongyao/Live264Streamer. — [Online; accessed 19-May-2022].

[32] jiaxin du. RTSPMultiCam: A RTSP server for streaming combined picture from multiple cameras based on FFMPEG and live555. — https://github.com/jiaxin-du/RTSPMultiCam. — [Online; accessed 19-May-2022].

[33] papan01. LiveStreamer: Android Screen Streaming - Live555 + MediaProject. — https://github.com/papan01/LiveStreamer. — [Online; accessed 19-May-2022].

[34] pedroSG94. rtmp-rtsp-stream-client-java: Library to stream in rtmp and rtsp for Android. All code in Java. — https://github.com/pedroSG94/rtmp-rtsp-stream-client-java/. — [Online; accessed 19-May-2022].

[35] pengliren. sms: SMS is a java-based streaming server. — https://github.com/pengliren/sms/. — [Online; accessed 19-May-2022].

[36] spex66. RTSP-Camera-for-Android: Android based RTSP Server which is able to serve live camera view to multiple RTSP clients, such as VLC. — https://github.com/spex66/RTSP-Camera-for-Android. — [Online; accessed 19-May-2022].