**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОМУ ПРОЕКТУ*

## *НА ТЕМУ:*

## *«Мультисерверный видеостриминг с множественными источниками»*

Студент __ИУ7-31М__ _____ ____Е. В. Брянская__
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент __ИУ7-31М__ _____ ____В. А. Иванов_ __
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ ____А.М. Никульшин_
(Подпись, дата) (И.О.Фамилия)

*2023 г.*

**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

# ЗАДАНИЕ
## на выполнение курсового проекта

по дисциплине Протоколы вычислительных сетей

Студенты группы ИУ7-31М

Брянская Екатерина Вадимовна
(Фамилия, имя, отчество)
Иванов Всеволод Алексеевич
(Фамилия, имя, отчество)

Тема курсового проекта Мультисерверный видеостриминг с множественными источниками

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) Кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

***Задание*** Разработать протокол для мультлисерверного стриминга. Проанализировать предметную область, выделить целевую аудиторию и сценарии его применения. Провести обзор существующих форматов видеорядов, обосновать выбор используемого формата в разрабатываемом протоколе. Определить участников видеостриминга и их функции. Определить способы их взаимодействия и составить набор соответствующих сообщений протокола и их содержание. Протокол должен поддерживать проверку целостности данных, повторный запрос потерянных и повреждённых кадров. Должен быть реализован механизм перераспределения трафика между источниками и управления скоростью передачи данных. Реализовать и протестировать клиент-серверное приложение, использующее данный протокол.

***Оформление курсового проекта:***

Расчетно-пояснительная записка на 20-30 листах формата А4.
Расчетно-пояснительная записка должна содержать постановку задачи, введение, аналитическую, конструкторскую, технологическую части, заключение и список литературы.

Дата выдачи задания «11» октября 2023 г.

| | | |
|---|---|---|
| **Руководитель курсового проекта** | _____ (Подпись, дата) | А.М.Никульшин (И.О.Фамилия) |
| **Студент** | _____ (Подпись, дата) | В.А.Иванов (И.О.Фамилия) |
| **Студент** | _____ (Подпись, дата) | Е.В.Брянская (И.О.Фамилия) |

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

По данным журнала Market Research Report [1] рынок видеостриминга сейчас оценивается более, чем 500 миллиардов долларов, и, предполагается, что к 2030 году эта сумма достигнет 1.9 триллионов долларов. Согласно статистике по всему миру насчитывается около 1.8 миллиарда подписок на сервисы потоковой передачи видео, примерно 26% пользователей которых признаются, что пользуются подпиской на постоянной основе не реже одного раза в неделю.

Потоковая передача видео в настоящее время более популярна и составляет более, чем 38.1% от общего объёма использования, чем кабельное или широковещательное телевидение, на долю которых приходится 30.9% и 24.7%.

Соответственно, ввиду непрекращающегося спроса на видеоплатформы, необходимо обеспечить эффективный способ взаимодействия многочисленных пользователей с этими сервисами для обеспечения наиболее качественной передачи информации.

Цель работы – разработать протокол для мультисерверного стриминга, обеспечивающий высокую производительность и масштабируемость.

Для достижения цели необходимо решить следующие задачи:

- проанализировать предметную область и выделить целевую аудиторию;
- определить сценарии взаимодействия участников передачи данных, составить набор соответствующих сообщений и их содержание;
- провести обзор существующих форматов видеорядов и обосновать выбор используемого формата в разрабатываемом протоколе;
- идентифицировать и сформулировать основные требования для мультисерверного стриминга;
- разработать протокол в соответствии с выделенным и требованиями;
- реализовать и протестировать прототип, позволяющий проверить правильность работы и эффективность протокола.

# 1 Аналитическая часть

## 1.1 Основные понятия

**Потоковый трафик** – тип трафика, для которого характерен просмотр и/или прослушивание информации по мере её поступления на конечное оборудование (мобильный телефон, компьютер, телевизор с доступом в Интернет и т.д.). [2] Основную часть потокового трафика составляет потоковое видео (или видеопоток).

**Видеостриминг** (или **стриминг видео**) – технология передачи видеоконтента через Интернет в режиме реального времени, при этом пользователь не должен ждать полной загрузки файла для просмотра. [3] Видео транслируется непрерывным потоком в виде последовательных кадров в специальном формате. Просмотр начинается в момент достаточной буферизации, обеспечивая при этом равномерное отображение данных.

Основой для передачи мультимедийной информации в настоящее время становятся мультисерверные платформы. Они способны отправлять данные разных типов и поддерживать трафик с различными характеристиками. Наибольшее распространение получили два варианта передачи потокового видео по сети.

- *Видео реального времени (real-time streaming)*, запись которого осуществляется одновременно с его просмотром, например, видеоконференции, прямые эфиры.
- *Видео по запросу* (progressive streaming). Предварительно записанные видеоряды хранятся на сервере, запрашиваются приложениями конечного пользователя и воспроизводятся при получении.

Привлечение мультипоточной загрузки, при которой данные загружаются сразу с нескольких серверов или источников, имеет следующие преимущества по сравнению с загрузкой только с одного сервера.

- *Увеличение скорости загрузки*. Мультипоточная загрузка позволяет ис-

пользовать полную пропускную способность нескольких серверов одновременно, ускоряя процесс передачи видеоконтента. Это особенно полезно при работе с большими файлами, например, с видео высокого разрешения.

- *Улучшение стабильности и отказоустойчивости*. Если один из источников недоступен или работает медленно по какой-то причине, то другие могут продолжать предоставлять необходимые данные. Это минимизирует возможные проблемы с недоступностью серверов.
- *Оптимизация использования сетевых ресурсов*. Разделение загрузки данных между несколькими серверами помогает избежать перегрузки одного сервера и эффективно использовать сетевые ресурсы.
- *Адаптация к сетевым ограничениям*. При привлечении нескольких серверов можно адаптировать процесс загрузки к изменениям сетевых условий (например, изменение скорости интернет-соединения), что помогает поддерживать стабильное и качественное воспроизведение.

Все эти преимущества делают мультисерверную загрузку более предпочтительной, особенно в случаях, если важна скорость передачи данных, стабильности и отказоустойчивость системы.

Как правило, система трансляции потоковых видео состоит из четырёх подсистем:

1) **Устройство кодирования** для сжатия видеопотока и загрузки его на медиасервер.
2) **Медиасервер**, отвечающий за хранение видеорядов и передачу пользователям. Он является ключевой единицей во всём процессе передачи. Основная его задача – взаимодействие с транспортной сетью при отправке пакетов в нужное время. Как правило, состоит из трёх компонентов механизма трансляции: *транспортный протокол, операционная система и система хранения*.
3) **Транспортная сеть**, которая транслирует пакеты от медиасервера до кли-

ентского устройства с помощью специально разработанных и стандартизированных протоколов. Последние обеспечивают такие услуги связи, как сетевая адресация, транспортировка и контроль за сеансом связи.

4) **Клиентское приложение**, декодирующее и воспроизводящее мультимедиапоток. Также опционально может присутствовать механизм синхронизации аудио и видео.

Действующий протокол между клиентом и сервером определяет:

1) синтаксис и формат данных

   Фиксируется чёткая структура сообщений или пакетов данных, которые передаются между устройствами. Как правило, это описание формата заголовков и тела сообщения.

2) способ установления соединения

   Протокол может определять процедуру процессов установления, поддержания и завершения соединения между устройствами.

3) основные операции и команды

   Описывается множество доступных операций, команд или запросов, которые могут быть выполнены в рамках существующего протокола.

4) обработку ошибок и контроль целостности данных

   Определяются методы и сценарии обработки некорректных ситуаций, способы контроля целостности данных.

5) управление потоком данных

   Может также описываться алгоритм регулирования скорости передачи потока информации.

В текущей работе будет рассматриваться протокол мультисерверного видеостриминга для передачи данных по запросу.

### 1.2 Возможные сферы применения и целевая аудитория

Подобный протокол может быть применён в рамках высоконагруженных систем, позволяя увеличивать скорость загрузки данных путём одновременно-

го запроса и получения разных частей контента с нескольких серверов. Такой подход позволяет не только ускорить загрузку видео, но и обеспечивает более плавное воспроизведение, поскольку некоторые части запрашиваются заранее и в случае потери, могут быть получены повторно.

Его привлечение в сервисах видеохостинга особенно полезно в случае нестабильного Интернет-соединения и большого объёма файла. Например, разрабатываемый протокол может быть привлечён в рамках обязательного внеурочного занятия в России, введённого в программы образовательных организаций начального, основного, среднего и профессионального образования. Каждый понедельник на первом уроке учащимся показывают видеофайлы, повествующие о наиболее актуальных событиях в стране, на основе которых строится дальнейшее обсуждение поднятых в них тем.

Таким образом, каждое утро понедельника классные руководители по всей стране скачивают подготовленные ролики с сайта «Разговоры о важном», создавая колоссальную нагрузку на серверы, из-за чего постоянно возникают сбои и сложности с загрузкой. Рассматриваемый в рамках текущей курсовой работы протокол может помочь решить некоторые возникающие проблемы, путём распределения процесса загрузки видеофрагментов по нескольким серверам.

Также его можно привлечь в процессе обмена видеофайлами между платформами для ускорения процесса переноса данных. Это может быть актуально для сервисов, размещающих видео сразу на нескольких платформах, например, YouTube, Rubube, VK Видео.

### 1.3 Форматы видеорядов

Наиболее часто встречающиеся форматы видео:

- **MP4 (по-другому MPEG-4 Part 14)** – формат, совместимый с большинством браузеров и поддерживаемый сайтами потокового видео, в частности, YouTube.
- **AVI (Audio Video Interleave)** – старый формат, разработанный Microsoft.

Поддерживается большинством популярных браузеров, работающих в системах Windows, Linux.

- **MPG, MPEG, MP2, MPE, MPV** – форматы, в которых обычно записывают видео, которые впоследствии не нужно будет редактировать.
- **MOV** – формат, разработанный Apple. Видео сохраняется в хорошем качестве, но файл занимает много места.
- **WebM** – формат, позволяющий получать видео небольшого размера среднего качества. Видео в таком формате подходят для YouTube и других сайтов потокового видео на платформе HTML5.

**УКАЗАТЬ КАКОЙ ФОРМАТ МЫ ИСПОЛЬЗУЕМ**

### 1.4 Основные этапы протокола мультисерверного стриминга с множественными источниками

На рисунке 1.1 схематично представлен общий алгоритм работы протокола. Ключевые моменты отмечены цифрами.

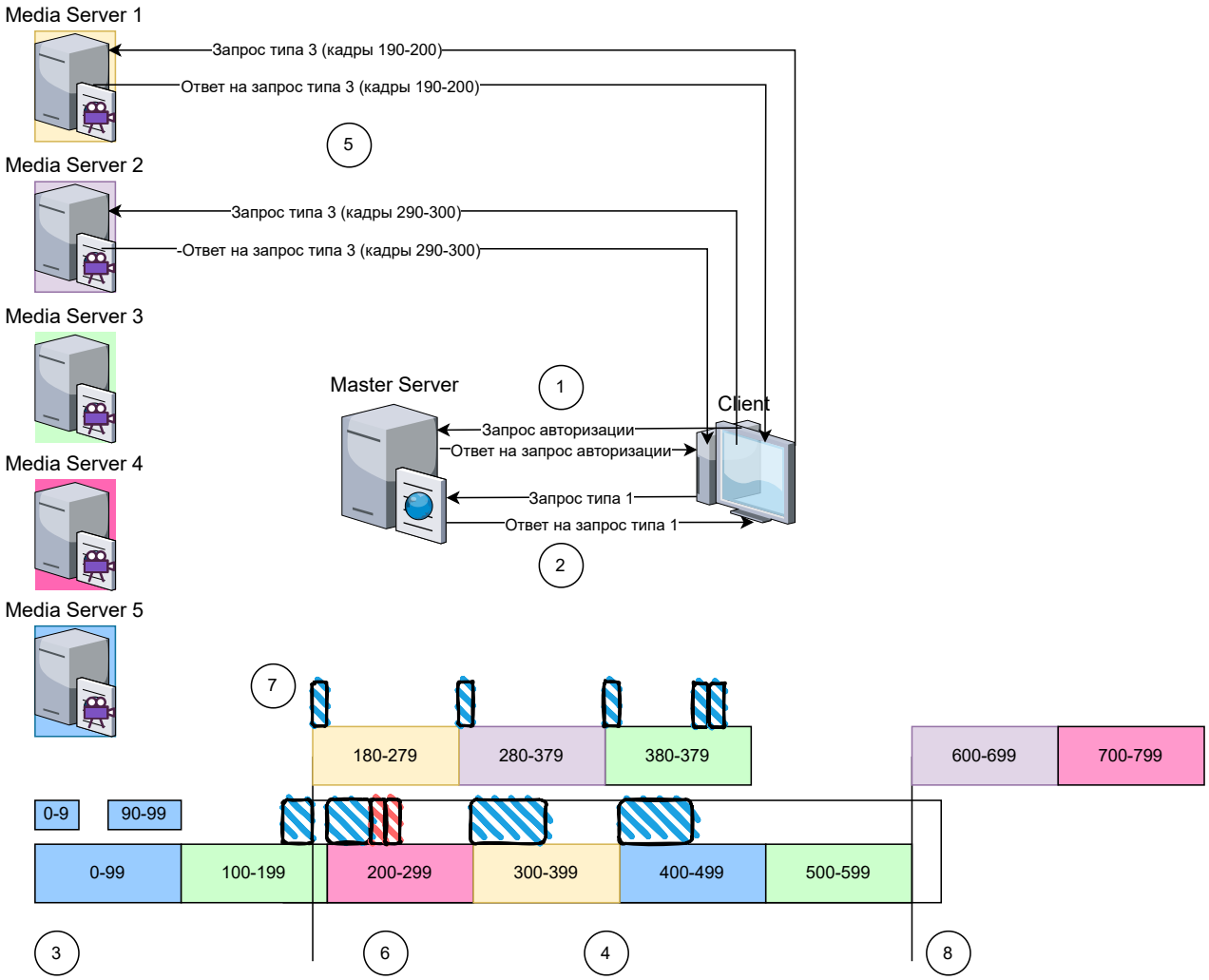


Рисунок 1.1 – Основные этапы протокола.

Весь видеофайл разбивается на несколько фрагментов (на рисунке метка 3), каждый из которых может быть получен от любого медиа-сервера.

Сначала клиент должен пройти процесс авторизации, обозначен цифрой 1, после успешного завершения которого он может запросить у мастер-сервера распределение интервалов видео по хостам (отмечен 2).

Получив его, клиент делает несколько параллельных запросов (отмечено как 5) на загрузку $n$ кадров.

В случае $m$ неуспешных запросов к какому-либо медиа-серверу (отмечено 6) клиент должен сообщить мастер-серверу о проблеме, инициировав перераспределение фрагментов между хостами.

Таким образом, клиент получает новый список медиа-серверов с перерас-

пределёнными видео-интервалами загружаемого файла (отмечено на рисунке цифрой 7) и начинает выгрузку ещё не полученных кадров.

Клиент загружает такой объём информации, который помещается в буфер (отмечен на рисунке цифрой 4).

Как только окно загрузки будет передвинуто на некоторую величину кадров, клиент должен отправить запрос на получение списка медиа-серверов для новых видео-фрагментов (цифра 8).

**Выводы**

Таким образом, в данной работе будет разрабатываться протокол для мультисерверного видеостриминга с множественными источниками. В качестве формата видео был выбран – вставить формат.

# 2 Конструкторская часть

## 2.1 Взаимодействие клиента и мастер-сервера

### 2.1.1 Общая схема

Основными компонентами в разрабатываемом протоколе являются клиент, мастер-сервер и медиасерверы. Сначала клиент устанавливает соединение с мастер-сервером, запрашивает у него список доступных для дальнейшего взаимодействия хостов.

В рассматриваемом протоколе взаимодействие между клиентом и мастер-сервером осуществляется по gRPC, между клиентом и медиа-серверами по UDP.

Общая схема обмена сообщениями между клиентом и мастер-серверов представлена на рисунке 2.1.

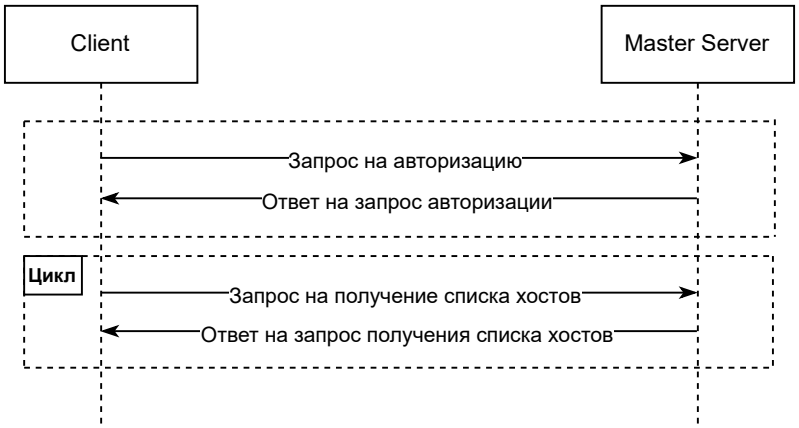


Рисунок 2.1 – Общая схема.

### 2.1.2 Авторизация

Первый шаг в установке соединения между клиентом и мастер-сервером – авторизация (рисунок 2.2).

Клиент отправляет авторизационный запрос в формате, представленном в таблице 1.

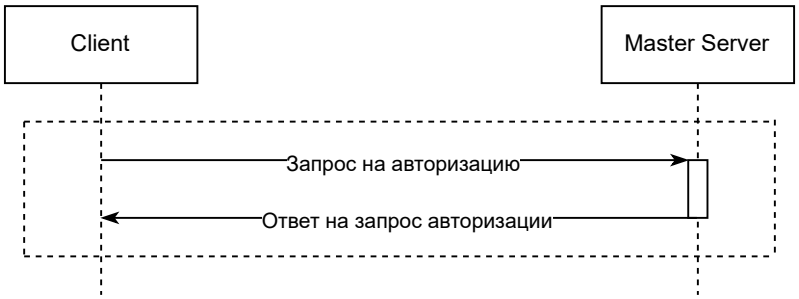


Рисунок 2.2 – Этап авторизации.

Таблица 1 – Атрибутивный состав запроса на авторизацию

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| username | string | Имя пользователя доменной учетной записи |
| password | string | Пароль пользователя доменной учетной записи |
| grant_type | string | Тип запроса на получение токена |
| client_id | string | Имя приложения |

В зависимости от полученных данных сервер либо передаёт токен доступа в ответном сообщении (таблица 2), либо возвращает сообщение об ошибке авторизации в формате из таблицы 3.

Таблица 2 – Атрибутивный состав ответа на успешный запрос авторизации

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| access_token | string | Токен доступа |
| expires_in_sec | integer | Число секунд, после которых токен перестанет быть валидным |
| token_type | string | Тип токена |

Таблица 3 – Атрибутивный состав ответа о непройденной авторизации на соответствующий запрос

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| error | string | Общее описание ошибки |
| error_description | string | Детальное описание ошибки |

### 2.1.3 Получение списка хостов

Далее в случае успешной авторизации клиент отправляет запрос на получение списка хостов (рисунок 2.3), к которым в последствии будут отправляться запросы на получение фрагментов видео.

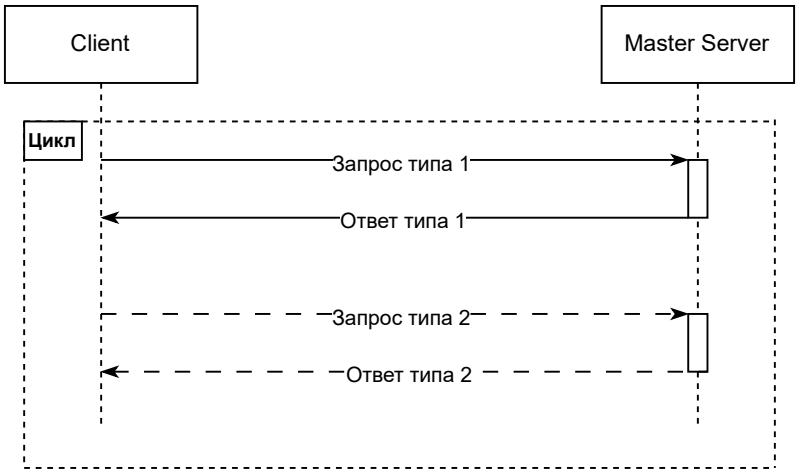


Рисунок 2.3 – Этап получения списка хостов.

Формат такого запроса представлен в таблице 4, на рисунке 2.3 обозначен как «Запрос типа 1».

Таблица 4 – Атрибутивный состав «Запроса типа 1»

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| access_token | string | Токен авторизации, указывается в заголовке запроса |

*Продолжение на следующей странице*

| Параметр | Тип | Описание |
|---|---|---|
| video_filename | string | Название видео-файла |
| start_time | timestamp | Временная метка начала временного окна |
| end_time | timestamp | Временная метка окончания временного окна |

В ответ мастер-сервер отправляет список хостов с указанием диапазона кадров («Ответ типа 1» на рисунке 2.3), которые можно с них загрузить, также дополнительно прилагается метаинформация о видеофайле (разрешение, длительность и т.д.). Атрибутивный состав этого сообщения приведён в таблице 5.

Таблица 5 – Атрибутивный состав ответа на «Запрос типа 1»

| Параметр | Тип | Описание |
|---|---|---|
| hosts | array[object] | Список хостов с указанием адреса и диапазона кадров |
| address | string | Адрес хоста |
| frame_start | integer | Номер первого фрагмента из диапазона |
| frame_end | integer | Номер последнего фрагмента из диапазона |
| metadata | object | Блок метаданных |
| resolution | string | Разрешение |
| codec | string | Кодек |
| duration | timestamp | Длительность |

Поскольку клиент запрашивает кадры видео только в пределах окна, то в случае его заполнения оно должно быть сдвинуто, и к мастер-серверу должен осуществляться повторный запрос на получение списка хостов («Запрос типа 1») для загрузки очередных фрагментов. Эта операция должна повторяться до

тех пор, пока не будет получен весь видеофайл.

В случае, если клиенту не удаётся загрузить $n$ частей видеофрагмента от одного и тоже же медиа-сервера в пределах текущего окна, то он должен отправить на мастер-сервер «Запрос типа 2», в котором указывается проблемный хост и временная метка последней успешно загруженной части видеофрагмента, загружаемого с данного медиа-сервера. Атрибутивный состав сообщения приведён в таблице 6.

Таблица 6 – Атрибутивный состав «Запроса типа 2»

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| access_token | string | Токен авторизации, указывается в заголовке запроса |
| video_filename | string | Название видео-файла |
| start_time | timestamp | Временная метка последней успешно загруженной части видеофрагмента |
| end_time | timestamp | Временная метка окончания временного окна |

Отправляя «Запрос типа 2» клиент инициирует повторную операцию определения подходящих для взаимодействия хостов. В ответ мастер-сервер отправляет новое перераспределение фрагментов видео между медиа-серверами, состав ответа приведён в таблице 7.

Таблица 7 – Атрибутивный состав ответа на «Запрос типа 2»

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| hosts | array[object] | Список хостов с указанием адреса и диапазона кадров |
| address | string | Адрес хоста |

*Продолжение на следующей странице*

| Параметр | Тип | Описание |
|---|---|---|
| frame_start | integer | Номер первого фрагмента из диапазона |
| frame_end | integer | Номер последнего фрагмента из диапазона |
| metadata | object | Блок метаданных |
| resolution | string | Разрешение |
| codec | string | Кодек |
| duration | timestamp | Длительность |

### 2.1.4 Получение видеофрагментов

Общая схема загрузки видеофрагмента показана на рисунке 2.4. После получения списка доступных хостов от мастер-сервера клиент отправляет сразу несколько запросов на загрузку частей фрагмента параллельно. Таким образом, осуществляется одновременное скачивание сразу нескольких частей видео-файла в рамках текущего окна загрузки.

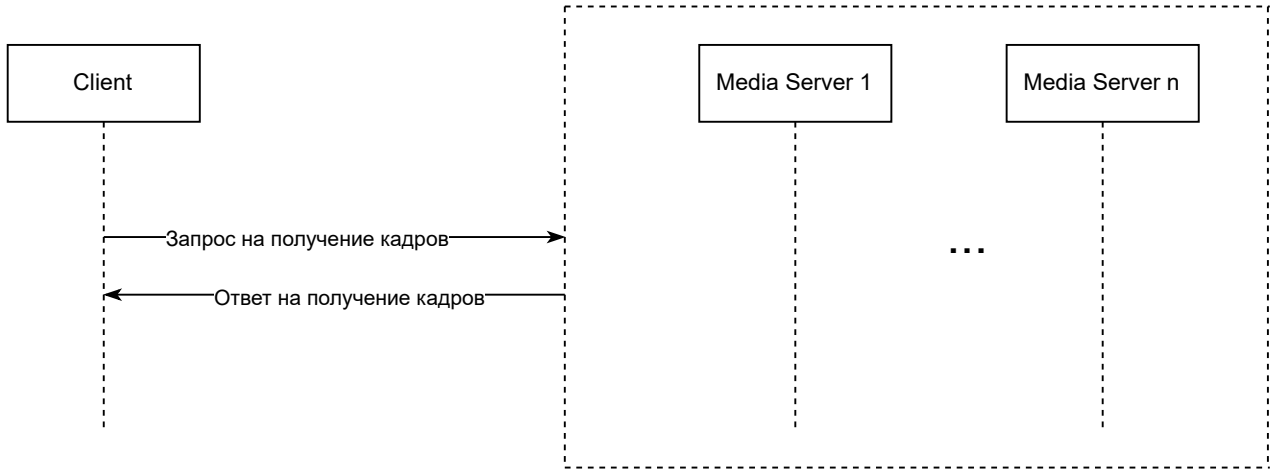


Рисунок 2.4 – Общая схема получения видеофрагментов.

На рисунке 2.5 представлена более детальная схема этого этапа. Параллельные запросы («Запрос типа 3» таблица 8) к медиа-серверам будут осуществляться до тех пор, пока не будет получена большая часть окна загрузки, после

чего оно динамически сдвигается, и клиент запрашивает у мастер-сервера список новых хостов.

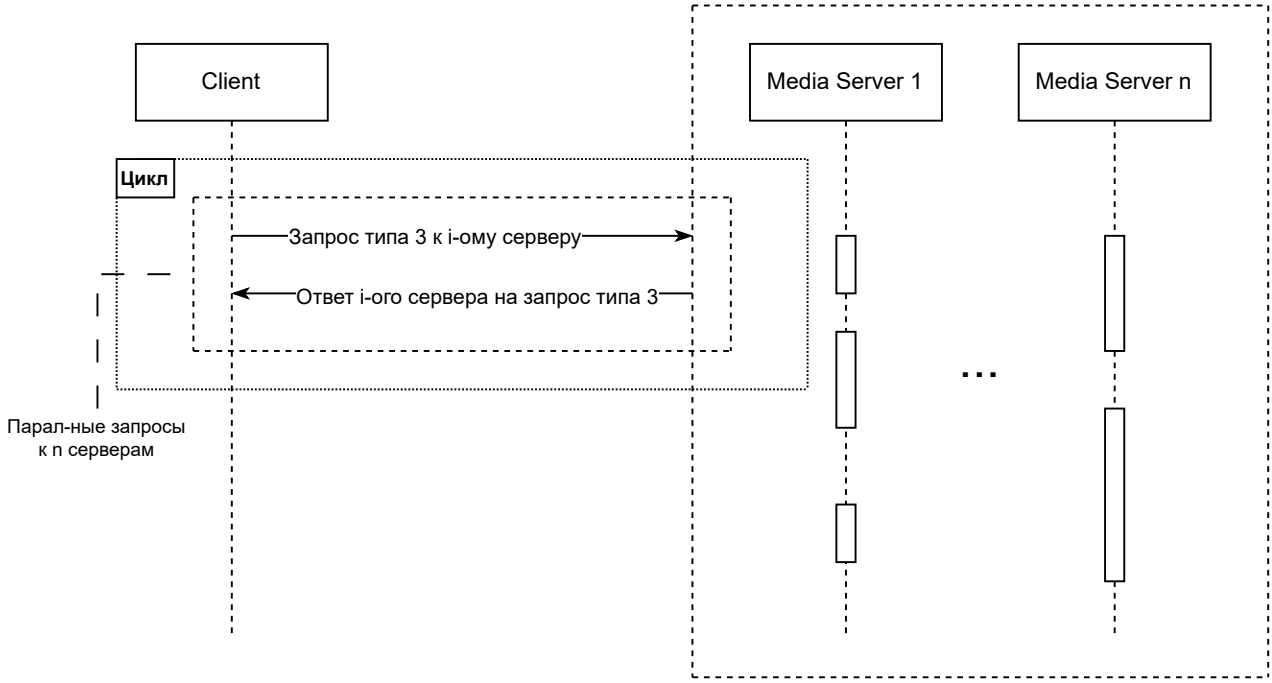


Рисунок 2.5 – Установка соединения.

Таблица 8 – Атрибутивный состав «Запроса типа 3»

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| access_token | string | Токен авторизации, указывается в заголовке запроса |
| video_filename | string | Название видео-файла |
| frame_start | integer | Номер первого фрагмента из диапазона |
| frame_end | integer | Номер последнего фрагмента из диапазона |

В ответ на поступивший запрос медиа-сервер формирует ответ, атрибутивный состав которого представлен в таблице 9.

Таблица 9 – Атрибутивный состав «Запроса типа 3»

| **Параметр** | **Тип** | **Описание** |
|---|---|---|
| num_frames | integer | Число передаваемых кадров |
| frames | array[object] | Список кадров |
| size | integer | Размер в байтах |
| frame | array[byte] | Кадр |

# 3 Технологическая часть

## 3.1 Выбор средств программной реализации

### 3.1.1 Основные средства

В качестве языка программирования был выбран Python 3 [4], ввиду нескольких причин.

- Также язык поддерживает объектно-ориентированный подход, что важно, поскольку в процессе реализации подразумевается использование этой методологии, позволяющей разрабатывать хорошо организованную и просто модифицируемую структуру приложения.
- Кроме того, предоставляются библиотеки для создания графического интерфейса, которые планируется использовать для отладки и наглядной демонстрации работы приложения.
- В дополнение, в процессе обучения был накоплен существенный опыт в использовании этого языка программирования.

В качестве среды разработки был выбран VSCode [5] в силу следующих факторов.

- Она бесплатна для студентов.
- Предоставляются удобные инструменты для написания, редактирования кода, а также графический отладчик.
- Помимо этого, является хорошо знакомой средой разработки, и какие-либо проблемы с взаимодействием сведены к минимуму, что позволяет сэкономить время.

### 3.1.2 Вспомогательные средства

Для сбора датасета использовалась кроссплатформенная система мгновенного обмена сообщениями Discord [**?**]. Это было сделано по следующим причинам.

- Она бесплатна для всех пользователей.

- Поскольку необходимо было опросить большое количество людей, встречаться лично было затруднительно, поэтому гораздо удобнее и проще организовать весь процесс дистанционно, что и позволяет сделать это приложение.
- Платформа очень популярна среди молодых людей, что доказывает недавнее исследование от апреля 2022 года [**?**]. Это упрощает поиск удобной большинству среды.
- Для этой системы возможно написание дополнительного API в виде discord-бота, который может выполнять различные задачи. Так, для ускорения процесса фиксации слов опрашиваемых было создано дополнительное приложение, которое в реальном времени переводит речь участников в текст, заносит всю информацию в файлы и по каждому человеку создаёт базу знаний.

### 3.2 Используемые библиотеки

Для разработки графического пользовательского интерфейса привлекалась библиотека PyQt5 [**?**]. Qt – один из самых популярных кроссплатформенных графических фреймворков [**?**], поэтому кроме документации, описано множество примеров его использования. Для наглядной разработки GUI привлекалась среда Qt Designer [**?**].

Для упрощения контроля над тем, правильно ли формируются графы и сети, используется библиотека NetworkX [**?**], позволяющая визуализировать подобные структуры.

В приложении и discord-боте для перевода речи пользователя в текст используется библиотека Speech Recognition [**?**]. Это инструмент от таких компаний, как Google, Microsoft, IBM и др. Для работы используется стандартный Google Speech API.

К другой библиотеке, Natasha [**?**], происходит обращение с целью выделения словосочетаний в предложениях. Natasha позволяет с помощью готовых правил решать базовые задачи NLP для русского языка такие, как:

- токенизация;
- сегментация;
- определение морфологических признаков;
- лемматизация/нормализация;
- выделение словосочетаний и т.д.

Кроме того, привлекается библиотека SciPy [**?**], которая ориентирована на работу с большим количеством данных, содержит много функций линейной алгебры, интерполяции, масштабирования данных.

**Выводы**

В данном разделе для реализации разрабатываемого метода выбран Python в качестве основного языка программирования, Javascript – как вспомогательный инструмент для реализации сопутствующего приложения для сбора датасета. В качестве среды разработки был выбран PyCharm.

Определены основные используемые библиотеки. Также изложен способ получения данных для статистического метода.

Кроме того, приведён и подробно описан интерфейс программы и продемонстрирована её работа. А также изложены основные пункты, по которым производилось тестирование ПО.

# ЗАКЛЮЧЕНИЕ

Таким образом, в рамках текущей

- 
- 
- 

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Video Streaming Market Size, Market Research Report [Электронный ресурс]. – Режим доступа: https://www.fortunebusinessinsights.com/video-streaming-market-103057 (Дата обращения: 02.12.2023)

2. Пакулова Екатерина Анатольевна Алгоритм распределения потокового трафика и трафика реального времени в гетерогенной беспроводной сети // Известия ЮФУ. Технические науки. 2014. №2 (151). URL: https://cyberleninka.ru/article/n/algoritm-raspredeleniya-potokovogo-trafika-i-trafika-realnogo-vremeni-v-geterogennoy-besprovodnoy-seti (дата обращения: 04.12.2023).

3. Apostolopoulos J. G., Tan W., Wee S. J. Video streaming: Concepts, algorithms, and systems //HP Laboratories, report HPL-2002-260. – 2002. – С. 2641-8770.

4. Документация по Python 3 [Электронный ресурс]. Режим доступа: https://docs.python.org/3/ (Дата обращения 01.12.2023)

5. Документация по Visual Studio Code [Электронный ресурс]. Режим доступа: https://code.visualstudio.com/docs (Дата обращения 01.12.2023)

# ПРИЛОЖЕНИЕ А

## Структуры ядра, содержащие информацию о сокетах