Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Название института

Работа допущена к защите
Должность руководителя М
_____ И.О. Фамилия
«_____» _____ 202X г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ДИПЛОМНАЯ РАБОТА БАКАЛАВРА
РАЗРАБОТКА СИСТЕМЫ АДАПТИВНОГО СТРИМИНГА ВИДЕО В НЕСКОЛЬКИХ СИНХРОНИЗИРОВАННЫХ ПОТОКАХ С РАСПРЕДЕЛЁННОЙ СЕГМЕНТИРОВАННОЙ ЗАГРУЗКОЙ КОНТЕНТА

по направлению подготовки 09.03.03 Прикладная информатика
Направленность (профиль) 09.03.03_YY Наименование направленности (профиля) образовательной программы

Выполнил
студент гр. 5130903/10301 В.Д. Ярцев

Руководитель
старший преподаватель,
степень, звание[1] Е.В. Комарова

Консультант[2]
должность, степень И.О. Фамилия

Консультант
по нормоконтролю[3] И.О. Фамилия

Санкт-Петербург
202X

[1]Должность указывают сокращенно, учёную степень и звание — при наличии, а подразделения — аббревиатурами. «СПбПУ» и аббревиатуры институтов не добавляют.

[2]Оформляется по решению руководителя ОП или подразделения. Только 1 категория: «Консультант». В исключительных случаях можно указать «Научный консультант» (должен иметь степень). Без печати и заверения подписи.

[3]Обязателен, из числа ППС по решению руководителя ОП или подразделения. Должность и степень не указываются. Сведения помещаются в последнюю строчку по порядку. Рецензенты не указываются.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние годы видеоконтент стал одним из основных источников потребления информации людьми. Технологии видеостриминга используются в различных сферах, включая бизнес, образование, медицину и индустрию развлечений, и играют в них большую роль. Видеоконтент зачастую используется менеджерами по продукции различных компаний для повышения внутренних ключевых показателей эффективности (KPI), включая важные продуктовые метрики: вовлечённость пользователей, удержание пользователей, брендовое восприятие. Всё это говорит о том, что видео является важным инструментом повышения конверсии восприятия людьми контента.

Решаемые при обработке и воспроизведении видео, являются достаточно нетривиальными из-за возникающей высокой сетевой и вычислительной нагрузки. В связи с чем для многих организаций в различных отраслях становится нерентабельным создание и поддержка собственных систем хостинга и стриминга видео и возникает потребность в использовании уже имеющихся решений.

Однако большинство современных сервисов видеостриминга не поддерживает воспроизведение нескольких синхронизированных потоков видео в одном экземпляре видеоплеера, хотя такая потребность возрастает с каждым годом, так как производимый пользователями контент становится технологически сложнее.

Актуальность

На текущий момент проблема синхронизации нескольких видеопотоков решается со стороны бизнеса-потребителя сервисов видеохостинга с помощью средств монтажа. Функциональная возможность переключения между несколькими синхронизированными видеопотоками позволит пользователю самостоятельно выбирать предпочтительные ему фрагменты контента, что положительно скажется на продуктовых и непродуктовых метриках потребления контента в различных сферах: медицина - улучшенный анализ сложных процедур и операций, образование - повышение эффективности обучения благодаря возможности сопоставления видеоматериала и презентации, субтитров, бизнес и сфера развлечений - повышение времени, проведённого пользователем (time spent).

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, ТЕХНОЛОГИЙ СТРИМИНГА, СЖАТИЯ И ПЕРЕДАЧИ ВИДЕОКОНТЕНТА ДЛЯ РЕАЛИЗАЦИИ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ

В данной главе проводится анализ предметной области работы, функциональности видеоплееров существующих стриминговых платформ, а также имеющихся решений в области хранения видеоконтента, и на основании полученных данных рассматриваются основные технологии для стриминга видео и области их применения, сжатия и кодирования видеоконтента с учётом совместимости с технологиями стриминга, а также протоколы для передачи файлов на разном уровне взаимодействия: между клиентом и сервером и межсерверном. Целью данной главы является определение оптимальных технологий для реализации системы с долгосрочным доступ к видеоконтенту и поддержанием возможности его адаптивного воспроизведения на клиентской стороне.

1.1. Анализ предметной области

1.1.1. Понятие видеостриминга

Цифровой мультимедийный контент охватывает широкий спектр цифровых данных, объединяющих текст, звук, изображения и видео, предоставляя пользователю разнообразные способы восприятия информации. С развитием информационных технологий цифровой контент стал важной частью жизни человека, включая интернет-платформы для хранения и распространения медиафайлов.

Одним из ярких примеров является видеоконтент, который благодаря широкому распространению Интернета, мобильных устройств и быстрому развитию технологий передачи данных стал доступен большому числу пользователей по всему миру [1]. С ростом спроса на видеоконтент возникла потребность в эффективных и удобных способах его доставки. Технологии видеостриминга, обеспечивающие бесперебойную передачу видео в реальном времени, стали одним из основных инструментов для решения этой задачи [1]. Эти технологии позволяют устранить необходимость в долгой загрузке файлов перед их воспроизведением, а также значительно облегчают доступ к контенту, позволяя обрабатывать контент на устройствах пользователей оптимально [2].

Видеостриминг делает возможным мгновенный доступ к видеоматериалам, что является важным фактором в эпоху, когда пользователи ожидают максимального удобства и скорости при получении информации.

Видеостриминг — это технология передачи видеоконтента в режиме реального времени или по запросу, позволяющая воспроизводить видеофайлы по мере их загрузки [1]. В отличие от традиционного скачивания, при котором необходимо дождаться полной загрузки файла перед его воспроизведением, видеостриминг обеспечивает более гибкий доступ к контенту. Это достигается путем разбиения видео на сегменты, которые последовательно передаются пользователю для декодирования и воспроизведения. Видеостриминг устраняет необходимость в загрузке файлов перед их просмотром, экономя время пользователей и пространство на их устройствах. Кроме того, видеостриминг решает проблему распределения контента. Он позволяет централизованно хранить видео на серверах и предоставить доступ к нему любому пользователю с подключением к интернету, устраняя географические ограничения [1].

1.1.2. Роль видеостриминга в различных сферах

Видеостриминг стал неотъемлемой частью различных сфер жизни, включая образование, бизнес, развлечения и социальные взаимодействия. Технологии потокового вещания предоставляют новые возможности для передачи информации в реальном времени и по запросу, что существенно влияет на коммуникацию и восприятие контента.

Роль видеостриминга в образовании

В образовательной сфере видеостриминг позволяет проводить лекции, семинары и тренинги в режиме реального времени, обеспечивая доступ к знаниям независимо от географического положения участников [3]. Это особенно актуально для дистанционного обучения и курсов повышения квалификации. Кроме того, видеостриминг дает возможность повысить вовлеченность студентов. Платформы, использующие видеостриминг, могут интегрировать различные элементы взаимодействия и навигации, которые помогают студентам повышать свою вовлечённость в учебный процесс. Более того, сам факт наличия функциональности навигации между видеоконтентом значительно увеличивает среднее время просмотра контента студентами [3], что говорит о необходимости их развития.

Роль видеостриминга в коммерческой сфере

В коммерческой сфере видеоконтент, включая видеостриминг, также способствует значительному увеличению вовлеченности аудитории. В отличие от статичных рекламных материалов, видеоконтент является динамичным и визуально привлекательным, что позволяет удерживать внимание зрителей на протяжении более длительного времени [4]. Видеоконтент позволяет создавать живые презентации, обзоры, инструкции, которые значительно эффективнее передают информацию, чем текстовые или статичные изображения. Это способствует лучшему пониманию продукта и повышает вероятность совершения покупки. Кроме того, 60% потребителей сначала обращают внимание на видеоконтент на веб-странице и только потом на текст или изображения. Также средний потребитель на 1200% больше делится видеоконтеном, чем ссылками и текстами вместе взятыми [4]. На рисунке видно, что видеоконтент удерживает больше всего внимания пользователей по сравнению с другими представлениями информации. Видеостриминг в данном случае играет роль инструмента обеспечения качественного просмотра контента.

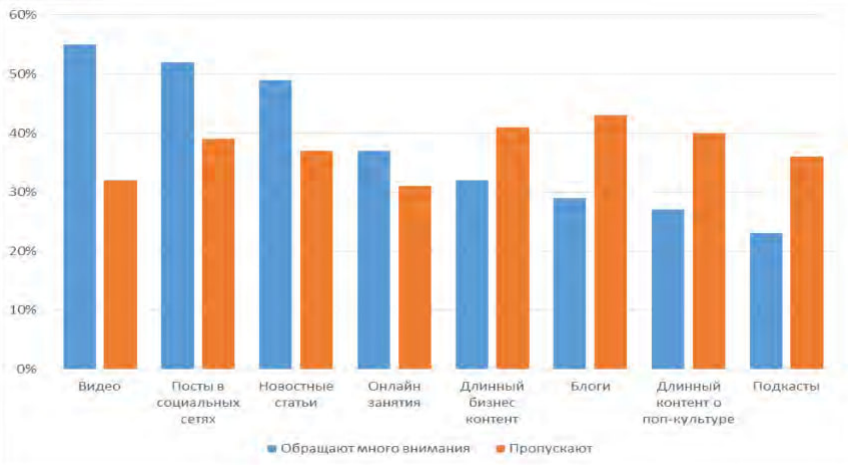

Рис.1.1. Распределение внимания пользователей между различными видами контента [4]

1.1.3. Основные концепции видеостриминга

Ключевая особенность видеостриминга — это возможность доставки и воспроизведения контента практически в реальном времени. Традиционно стриминг используется для передачи как предварительно закодированных видеофайлов, так и живых трансляций. Однако существует ряд концепций видеостриминга, которые необходимо учитывать при разработке для обеспечения качественного воспроизведения контента [1]:

- Сжатие видео (Video Compression): видео требует больших объемов данных для хранения и передачи, что создает сложности при передаче через сети с ограниченной пропускной способностью, особенно для пользователей с медленным интернет-соединением. Сжатие видео позволяет уменьшить размер данных, сохраняя при этом приемлемое качество изображения;
- Адаптивный битрейт (Adaptive Bitrate Streaming): сетевые условия могут меняться в реальном времени, что приводит к снижению качества видео, если пропускная способность сети не соответствует требованиям для передачи высокого качества видеопотока. Эта концепция позволяет автоматически изменять качество видеопотока в зависимости от текущих сетевых условий;
- Протоколы потоковой передачи (Streaming Protocols): чтобы эффективно передавать видеопоток, необходимо синхронизировать передачу данных между сервером и клиентом, а также обеспечить управление потоком и контроль за его состоянием. Протоколы, соответствующие этому требованию, позволяют доставлять видео и аудио с минимальными задержкам при поддержки высокого качества передачи;

Соблюдение данных концепций является необходимым минимумом для реализации системы стриминга видео с высоким качеством.

1.1.4. Перспективы развития платформ видеостриминга

Технологии видеостриминга продолжают стремительно развиваться, и с каждым годом появляются новые возможности для улучшения качества контента, удобства пользователя и функциональности платформ. В то время как основное внимание традиционно уделяется качеству видеопотока, в последние годы наблюдается значительный сдвиг в сторону улучшения интерфейсов и функционала, который может значительно повысить вовлеченность зрителей и их удовлетворенность от использования платформ. Современные системы стриминга должны не только обеспечивать высокое качество видео, но и фокусироваться на улучшении интерфейсной функциональности. Это включает в себя такие аспекты, как удобство навигации, интерактивные возможности и персонализация контента. Разнообразие пользовательских сценариев, которые предлагает видеостриминг, способствует более активному вовлечению зрителей, а значит, и повышению их времени, проведенного на платформе.

Одним из перспективных направлений, которое рассматривается в ходе этой работы, является внедрение возможности одновременного воспроизведения нескольких синхронизированных видеопотоков, между которыми зрители могут переключаться по своему усмотрению. Например, в спортивных трансляциях или в образовательных видеокурсах пользователи смогут переключаться между разными углами камеры, следить за несколькими персонажами или событиями одновременно. Такая функциональность предоставляет зрителям больше контроля над тем, что и как они хотят наблюдать, делая просмотр контента более персонализированным. Важно подчеркнуть, что эта клиентская функциональность требует соответствующих технологий для стриминга видео по запросу на стороне сервера. В отличие от традиционного потокового видео, где контент передается в реальном времени, видео по запросу позволяет пользователю выбирать и просматривать видео в любое время.

1.2. Анализ функциональности видеоплееров существующих стриминговых платформ

Стриминговые платформы активно развиваются, в том числе их технический и интерфейсный функционал. Быстрое развитие возможностей видеоплееров стремительно повышает качество взаимодействия с пользователями, что приводит к увеличению вовлеченности зрителей. Эта вовлеченность является ключевым элементом для монетизации контента, через рекламу или другие механизмы, напрямую влияя на прибыль стриминговых сервисов [5].

Для анализа существующих стриминговых платформ стоит рассматривать те, которые имеют наибольший срок эксплуатации и стабильный спрос. Это обусловлено тем, что они не только предлагают проверенные временем технологии, но и формируют тренды в области функционала, которые улучшают пользовательский опыт. Среди таких платформ можно выделить YouTube и Twitch, которые являются лидерами в своей категории и на протяжении многих лет задают стандарты в области потокового видео. Из базовых функций, которые содержат плееры всех перечисленных платформ можно выделить:

1. Управление воспроизведением:
    - Кнопки воспроизведения и паузы: стандартные кнопки для начала и приостановки воспроизведения видео;

- Перемотка по времени: возможность перематывать видео по временной шкале плеера. Прокрутка осуществляется с точностью до нескольких секунд, что позволяет быстро перемещаться по видео;
- Управление громкостью: встроенная регулировка громкости с помощью ползунка, а также кнопка для выключения звука.

2. Полноэкранный режим: возможность развернуть видео на весь экран, что улучшает восприятие контента;
3. Регулировка скорости воспроизведения: плеер дает возможность замедлять или ускорять видео, регулируя скорость воспроизведения (например, 0.5x, 1x, 1.5x, 2x);
4. Поддержка доступности и отзывчивого интерфейса: плеер адаптирован под различные устройства, включая мобильные телефоны, планшеты и компьютеры. На мобильных устройствах интерфейс становится более удобным для управления с помощью жестов. Кроме того, управление плеером доступно также и с клавиатуры и других устройств ввода.

Из платформенных особенностей можно выделить наличие у плеера YouTube субтитров на разных языках, созданных предварительно с помощью технологий машинного обучения, а также график интерактивной популярности, характеризующий, какие части видео пользователи чаще всего просматривают, перематывают или возвращаются к ним. Платформа Twitch, помимо перечисленного выше функционала, содержит интерактивные виджеты для проведения опросов разного типа, однако такая функциональность актуальна только для видеостриминга в режиме реального времени, так как выполняет роль коммуникации между автором и пользователями в процессе производства и потребления контента. Интерфейс видеоплееров рассмотренных платформ представлен на рисунках 2-3. Все перечисленные платформы предоставляют пользователям схожие базовые возможности для работы с видеоконтентом. Однако создание контента становится всё более сложным процессом, поскольку авторы действуют в условиях высокой конкуренции и вынуждены разрабатывать новые подходы к его производству. Сегодня популярным становится использование нескольких камер или видеопотоков, особенно в таких сферах, как образование, медицина или индустрия развлечений и медиа, где требуется передавать пользователю одновременно несколько видеопотоков, например, для отображения полной картины о проведении хирургической операции. Тем не менее, текущие платформы предоставляют возможность воспроизведения

контента преимущественно в одном потоке. Управление отображением нескольких элементов видео осуществляется авторами посредством монтажа. Это ограничение накладывает дополнительные требования на процесс создания видео и не всегда позволяет в полной мере отразить потенциал современных методов производства контента. Для удовлетворения потребностей таких авторов и улучшения пользовательского опыта внедрение поддержки воспроизведения видео в нескольких синхронизированных потоках на уровне платформы позволило бы повысить вовлечённость пользователей за счёт большего выбора и гибкости в просмотре контента. В связи с этим в рамках данной работы будет разработано приложение, реализующее эту функциональность. Однако для реализации таких возможностей на клиентской стороне необходимо также, чтобы на стороне сервера поддерживались адаптивность стриминга, технологии сжатия и хранения видео, а также возможность сегментированной загрузки контента для авторов. Эти требования обеспечивают качественное воспроизведение видео для пользователей, даже при низкой пропускной способности сети, и для авторов контента, так как позволяют целостно загружать видео без потери данных и с возможностью восстановления сессий загрузки.

1.3. Анализ существующих решений в области хранения видеоконтента

Хранение больших объемов бинарных данных (blob) при разработке системы с распределённой загрузкой контента является технически нетривиальной задачей, требующей эффективного управления большими объемами данных. Важно учитывать вопросы масштабируемости, скорости доставки и безопасности. Для решения этих проблем необходимо провести анализ современных решений в этой области и заимствовать лучшие практики, чтобы создать оптимальную архитектуру системы.

Amazon Web Services

Amazon web Services (AWS) - один из старейших игроков на рынке облачных технологий, основанный в 2006 году. Он предоставляет широкий спектр компьютерных услуг, таких как облачные хранилища, службы баз данных, аналитика, сетевые сервисы и многое другое [6].

AWS состоит из компонентов перечисленных ниже:

- Amazon CloudFront: этот сервис предназначен для доставки контента, используемого на веб-сайтах. Он поддерживает как статические, так и динамические данные, а также потоковое видео. CloudFront использует

распределенную сеть серверов по всему миру для оптимизации доставки контента пользователям;

- Балансировка нагрузки (Load Balancing): балансировка нагрузки позволяет повысить эффективность работы серверов и приложений. Это важный элемент сетевой инфраструктуры, который улучшает архитектуру классических веб-приложений. В AWS трафик распределяется между экземплярами сервисов, размещенными на разных ресурсах. При этом нагрузка также корректируется в зависимости от добавления или удаления экземпляров сервиса из балансировщика;
- Управление безопасностью (Security Management): этот компонент обеспечивает безопасность с помощью групп безопасности, которые функционируют как фильтры для входящего трафика. Они позволяют задавать параметры доступа, такие как порты, протоколы и диапазоны IP-адресов, ограничивая доступ к экземплярам сервисов на основе заданных правил;
- Amazon RDS (Amazon Relational Database Service): сервис Amazon RDS предоставляет возможность работы с реляционными базами данных, аналогичными MySQL или Microsoft SQL, обеспечивая доступ к базе данных в облаке с удобным управлением и настройкой;
- Эластичный кэш (Elastic Cache): система управления кэшированием в облаке, которая обеспечивает эффективную работу за счет сокращения нагрузки на серверы. Она помогает оптимизировать использование памяти и снижать нагрузку на сервисы, улучшая их надежность;
- Простой сервис очередей (Simple Queue Service): облачный сервис для асинхронного обмена сообщениями между компонентами распределенных систем. Обеспечивает надежную доставку, масштабируемость, поддержку стандартных и FIFO очередей, а также интеграцию с другими сервисами AWS. Позволяет снизить связность компонентов, оптимизировать производительность и управлять нагрузкой на системы.

Яндекс.Облако

Яндекс.Облако — это набор взаимосвязанных сервисов и инструментов для работы в облачной платформе [7]. Это платформа является отечественной и предоставляет для клиентов сервисы и инструменты, решающие те же проблемы, что и составляющие части AWS. Яндекс. Облако состоит из компонентов перечисленных ниже [8]:

- Объектное хранилище (Object Storage): этот сервис представляет собой масштабируемое хранилище для объектов данных, которое подходит для хранения любых типов файлов;
- Управляемые базы данных (Managed Databases): этот компонент включает поддержку PostgreSQL, MySQL, MongoDB, Redis, ClickHouse, Elasticsearch и других баз данных. Это решение похоже на AWS RDS;
- Load Balancers (Network/Application): Балансировщики нагрузки для распределения трафика между серверами, аналогичные AWS Load Balancing;
- Очереди сообщений (Message Queue): сервис позволяет организовать асинхронный обмен сообщениями между компонентами системы, аналогичен Amazon SQS.

И Yandex.Cloud, и AWS предоставляют мощные и гибкие решения для хранения, обработки и передачи больших бинарных объектов, которые обеспечивают высокую надежность, масштабируемость и интеграцию с другими сервисами. Однако их использование несёт в себе ряд определённых рисков и ограничений при разработке рассматриваемой в ходе работы системы, так как обе платформы работают на основе подписки или оплаты за использование, что делает их применение оправданным только для проектов с достаточным финансовым бюджетом, оба решения являются проприетарными и не предоставляют доступа к исходному коду, что ограничивает гибкость настройки и прозрачность. В связи с этим для решения поставленных в ходе работы задач оптимальным выбором будет реализовать платформу для хранения контента самостоятельно на основе следующих компонентов: балансировщики нагрузки, очереди сообщений, объектное хранилище, управляемые базы данных, управляемые базы данных для кэширования.

1.4. Технологии стриминга видео

Стриминг позволяет передавать видеоконтент в виде пакетов данных, что обеспечивает воспроизведение мультимедиа ещё до завершения загрузки файла. Такой подход снижает задержки перед началом просмотра и уменьшает требования к объему хранилища на стороне пользователя [1].

Стриминг позволяет эффективно распределить сетевую и вычислительную нагрузки не только на стороне клиента, так как производит загрузку только необходимого в текущий момент времени пользователю сегмента контента, но и на стороне сервера, где нет необходимости обрабатывать и передавать весь объём

данных сразу. Кроме того, использование технологий стриминга хорошо согласуется с микросервисной архитектурой приложений и позволяет без затруднений горизонтально масштабировать раздачу видеоконтента - увеличивать количество реплик сервисов, отвечающих за хранение и передачу контента пользователям при соответствующих запросах.

Технологии стриминга можно разделить на следующие категории [1]:

- Способ кодирования:
    - В реальном времени (Real-time encoding);
    - Предварительное (Pre-encoded video).
- Тип связи:
    - Точка-точка (Point-to-point);
    - Многовещание (Multicast);
    - Широковещание (Broadcast).
- Скорость передачи данных:
    - Постоянная (Constant Bit Rate, CBR);
    - Переменная (Variable Bit Rate, VBR).
- Описание потока:
    - Одиночное описание (Single Description Coding, SD);
    - Множественное описание (Multiple Description Coding, MD).
- Протокол передачи данных:
    - TCP;
    - UDP.
- Тип стриминга:
    - Прямой эфир (Live streaming);
    - Видео по запросу (On-demand streaming).
- Адаптивность:
    - Адаптивный стриминг (Adaptive streaming);
    - Неадаптивный стриминг (Non-adaptive streaming).

Технологии стриминга видео, которые будут использованы в системе, определяются на основе решаемых задач, так как каждая из них соответствует определённым критериям. Иными словами, выбор технологий должен опираться на их соответствии этим критериям для обеспечения наиболее эффективного решения. Основными критериями для реализации рассматриваемой в ходе работы системы являются:

- Способ кодирования: предварительное - система должна поддерживать долгосрочный доступ к видеоконтенту, а не доступ к контенту в реальном времени;
- Тип связи: точка-точка - в системе связь осуществляется между двумя конечными точками — сервером и клиентом;
- Скорость передачи данных: переменная - каждый видеопоток должен быть адаптивен и менять битрейт в зависимости от динамичности контента;
- Описание потока: одиночное - видеопоток может быть представлен в виде одного файла со списком сегментов, каждый из которых может быть получен клиентом по запросу;
- Протокол передачи данных: TCP - система предоставляет долгосрочный доступ к контенту, все сегменты видео хранятся на сервере и являются целостными, поэтому должны доставляться пользователю без потерь;
- Тип стриминга: видео по запросу;
- Адаптивность: адаптивный стриминг - система должна иметь возможность регулировать качество воспроизведения видео на стороне пользователя в зависимости от индивидуальной скорости передачи данных.

1.4.1. Протокол RTP

RTP предоставляет функции транспортировки данных в реальном времени от источника до получателя, подходящие для приложений, передающих данные в реальном времени, такие как аудио, видео или данные моделирования, через мультикаст или юникаст-сети [9].

Мультикаст-сети (Multicast) используются для отправки одного потока данных нескольким получателям одновременно.

Юникаст-сети (Unicast) предполагает индивидуальную передачу данных от отправителя к каждому получателю. Для каждого клиента создаётся уникальный поток.

Каждый RTP-пакет содержит фиксированный заголовок, за которым следует полезная нагрузка переменной длины. Заголовок включает информацию, необходимую для синхронизации и демультиплексирования медиапотоков [9]. Формат заголовка представлен на рисунке 1.2.

RTP использует переменную скорость передачи данных в зависимости от характеристик медиа и сети. Этот протокол также обычно использует одно описание

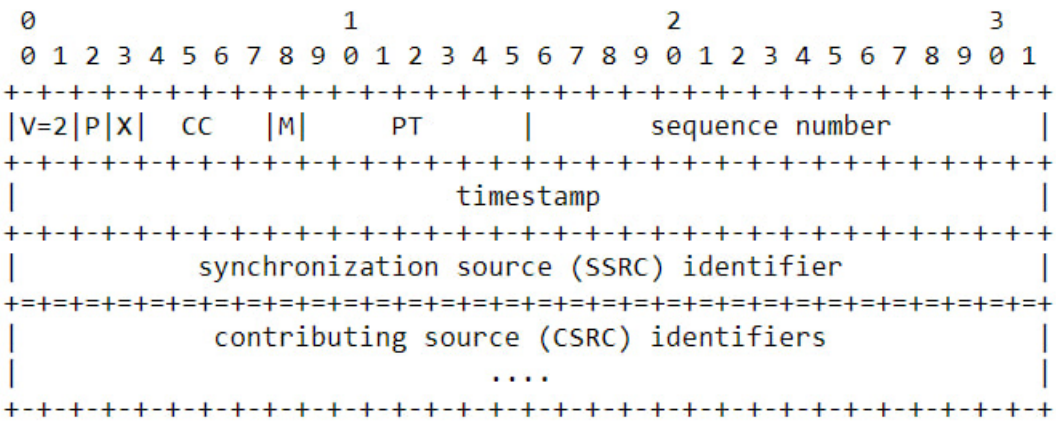

Рис.1.2. Заголовок RTP-пакета [9]

потока для мультимедийных данных, где каждый поток определяется отдельно для синхронизации. В качестве протокола для передачи данных используется UDP, так как он предназначен для приложений с низкими задержками и реального времени - целостность данных не играет ключевой роли, допустимы небольшие потери пакетов. RTP поддерживает адаптивный стриминг с использованием RTCP, который предоставляет отчеты о состоянии сети. Это позволяет адаптировать видеопоток [9].

1.4.2. Стандарт MPEG-DASH

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) — это стандарт для эффективной потоковой передачи мультимедийных данных с использованием существующей инфраструктуры HTTP. Он поддерживает как потоковую передачу по запросу (on-demand), так и в реальном времени (live streaming) [10].

Этот стандарт разделяет видео и аудио контент на небольшие сегменты, каждый из которых может быть закодирован с разным битрейтом. Также он поддерживает возможность определения длины сегмента по времени, что крайне удобно для выбора оптимального значения при запуске, например, A/B экспериментов для приложений, использующих эту технологию. Без использования средств нагрузочного тестирования - только на основе различий в продуктовых и технических метриках между двумя версиями приложения можно определить оптимальное значение длины сегмента.

Воспроизведение начинается с загрузки первого сегмента, после чего плеер динамически выбирает сегменты с оптимальным качеством в зависимости от текущей скорости сети и возможностей устройства. MPEG-DASH не зависит от кодеков, что позволяет использовать различные форматы сжатия: H.264, H.265,

VP8 и другие. Стандарт использует в качестве протокола передачи данных на прикладном уровне HTTP, на транспортном - TCP [11].

Важно отметить, что использование протоколов HTTP и TCP в основе стандарта позволяет поддерживать целостность передаваемых пользователю сегментов, что влечёт за собой определённые накладные расходы при передаче данных, но лучше всего подходит для стриминга видео по запросу, так как улучшает пользовательский опыт в системах с долгосрочным доступом к контенту.

Стандарт поддерживает как потоковое вещание по запросу, так и прямые трансляции (live streaming), включая функции перемотки назад и паузы в прямом эфире. Представление видеопотока описывается в файле формата MPD. Файл MPD (Media Presentation Description) содержит структуру мультимедийного контента, включая информацию о сегментах, их длительности, битрейте, разрешении и других параметрах. MPD может быть статическим или динамическим, в зависимости от типа контента [11]. Структура формата MPD представлена на рисунке 1.3.

1.4.3. Протокол HLS

HTTP Live Streaming (HLS) — это протокол адаптивной потоковой передачи мультимедийного контента, разработанный компанией Apple, который используется для доставки мультимедийного контента по сети Интернет [12].

HLS поддерживает потоковую передачу в реальном времени, где данные кодируются и сегментируются для прямых трансляций сразу при получении данных и передаёт их пользователю.

Этот протокол также поддерживает передачу заранее закодированных видео по запросу пользователя. Предварительно созданные и закодированные сегменты упрощают доставку пользователю и минимизируют задержки. Основной режим работы HLS — это точка-точка, где клиент напрямую запрашивает сегменты через HTTP у сервера. Протокол изначально не поддерживает многовещание или широковещание [12]. Для работы с такими режимами требуется дополнительная инфраструктура (например, использование CDN).

HLS позволяет использовать переменный битрейт. Клиент адаптируется к доступной пропускной способности, выбирая поток с соответствующим качеством. Это позволяет оптимизировать использование сети. Протокол также теоретически поддерживает постоянный битрейт, однако особое внимание в протоколе уделяется именно переменному битрейту. HLS работает исключительно через HTTP, а

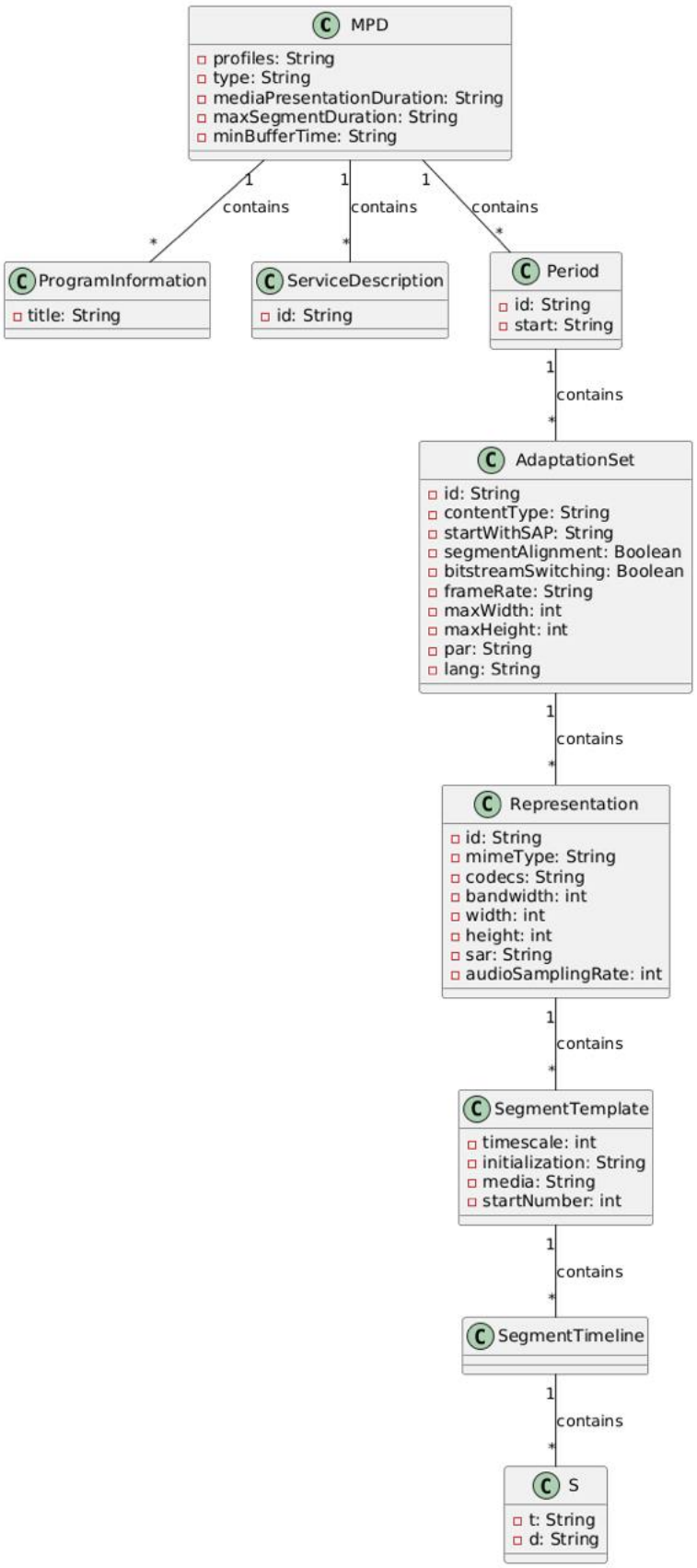

Рис.1.3. Структура формата MPD [9]

значит, поверх TCP. Это обеспечивает целостность данных при передаче [12]. HLS использует единую последовательность сегментов, которые представляют поток

на определённом уровне качества [13]. Его можно отнести к одиночному типу описания потока.

Так как протокол был создан компанией Apple, он занимает доминирующее положение в её продуктах, в частности, в мобильной и веб версиях браузера Safari. HLS поддерживается по умолчанию в этом браузере и не требует установки дополнительной инфраструктуры для использования, что выделяет её среди остальных технологий, когда речь идёт о продуктах Apple.

Концептуально HLS состоит из трёх частей: серверного компонента, компонента раздачи и клиентского программного обеспечения [13]. Архитектура протокола представлена на рисунке 1.4.

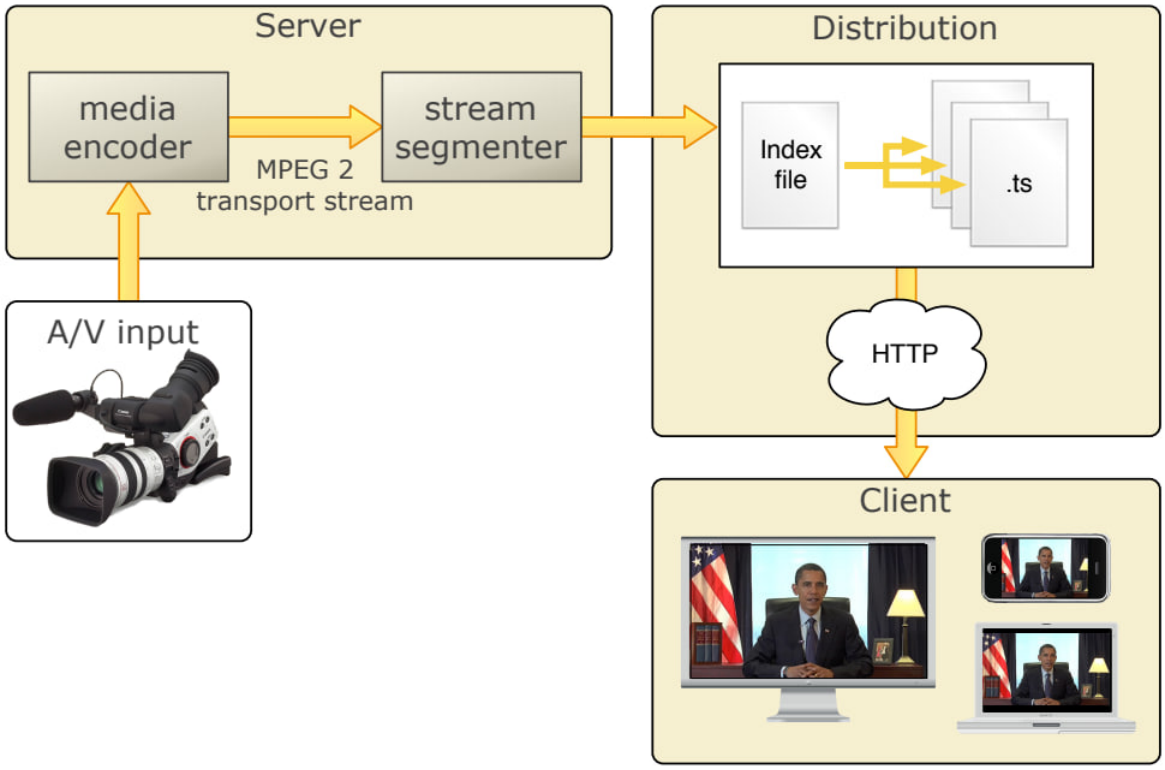

Рис.1.4. Архитектура протокола HTTP Live Streaming [13]

1.4.4. Сравнительный анализ технологий стриминга видео

Для выбора оптимального решения стриминга видео с долгосрочным доступом к контенту необходимо сравнить определённые ранее технологии по выделенным в разделе 1.2 критериям. Рассмотрим таблицу 1.1.

Таблица выше показывает, что для долгосрочного доступа к видеоконтенту наиболее подходящими решениями являются HLS и MPEG-DASH. Обе технологии поддерживают переменный битрейт, благодаря чему возможен адаптивный

Таблица 1.1

Сравнение технологий стриминга видео

Критерий сравнения	RTP	MPEG-DASH	HLS
Способ кодирования	Реальное время (Real-time)	Предварительное (Pre-encoded) и Реальное время (Real-time)	Предварительное (Pre-encoded)
Тип связи	Точка-точка (Point-to-point), Многовещание (Multicast)	Точка-точка (Point-to-point)	Точка-точка (Point-to-point)
Скорость передачи данных	Переменная (VBR)	Переменная (VBR), Постоянная (CBR)	Переменная (VBR), Постоянная (CBR)
Описание потока	Множественное (MD)	Одиночное (SD)	Одиночное (SD)
Протокол передачи данных	UDP	TCP	TCP
Тип стриминга	Прямой эфир	Видео по запросу, Прямой эфир	Видео по запросу, Прямой эфир
Адаптивность	Неадаптивный стриминг	Адаптивный стриминг	Адаптивный стриминг

стриминг. Они используют протокол TCP, что гарантирует целостность данных при передаче. Эти протоколы позволяют получить видео по запросу и обеспечивают качественный пользовательский опыт на разных устройствах.

В качестве основной технологии для стриминга видео явно стоит выделить MPEG-DASH. Помимо остальных преимуществ, эта технология позволяет более качественно минимизировать задержки при передаче данных, благодаря более атомарному разбиению видео на сегменты [14]. Однако для разных версий браузера Safari стоит отдать предпочтение протоколу HLS. Инфраструктура этого браузера оптимизирована именно для работы с HLS. Таким образом, для поддержки кроссплатформенности рассматриваемой в ходе работы системы наилучшим решением будет использовать и MPEG-DASH, и HLS, но для разных браузерных окружений. Загрузку инфраструктуры MPEG-DASH необходимо производить только при отсутствующей поддержке HLS.

1.5. Технологии сжатия видеоконтента

При работе с видео неизбежно возникновение проблемы хранения и передачи видеофайлов, которые связаны с их большими размерами. Это значительно затрудняет эффективную обработку и передачу данных по сетям с низкой пропускной способностью. Для уменьшения размера видео используются кодеки, которые применяют различные алгоритмы сжатия данных. Кодек - это программное обеспечение, предназначенное для преобразования данных определенного формата с целью их хранения, передачи или шифрования [15]. В данной главе будет рассмотрен ряд кодеков, которые могут быть использованы для эффективного решения этой проблемы.

1.5.1. Видеокодек H.264

Кодек H.264 или AVC (Advanced Video Coding) - это один из наиболее широко используемых стандартов сжатия видео. H.264 разработан таким образом, чтобы обеспечить баланс между высоким качеством видео и эффективностью сжатия [16]. Особенности H.264/AVC:

- Эффективные алгоритмы сжатия: В H.264 используются различные технологии для достижения высоких степеней сжатия при сохранении качества видео. К ним относятся:
    - Межкадровое предсказание: алгоритм позволяет уменьшить избыточность информации между кадрами путем обращения к ранее закодированным кадрам [16];
    - Внутрикадровое предсказание: этот метод минимизирует избыточность в пределах одного кадра за счет прогнозирования значений пикселей на основе соседних пикселей [16]. Режимы внутрикадрового предсказания представлены на рисунке 1.5.
- Адаптивное управление битрейтом: кодек позволяет определять битрейт видео или сегмента видео в зависимости от сложности контента [16].

Преимущества H.264:

- Высококачественное видео при низких скоростях передачи: кодек H.264 хорошо себя показывает в получении высококачественного видео при небольшой пропускной способности сети;

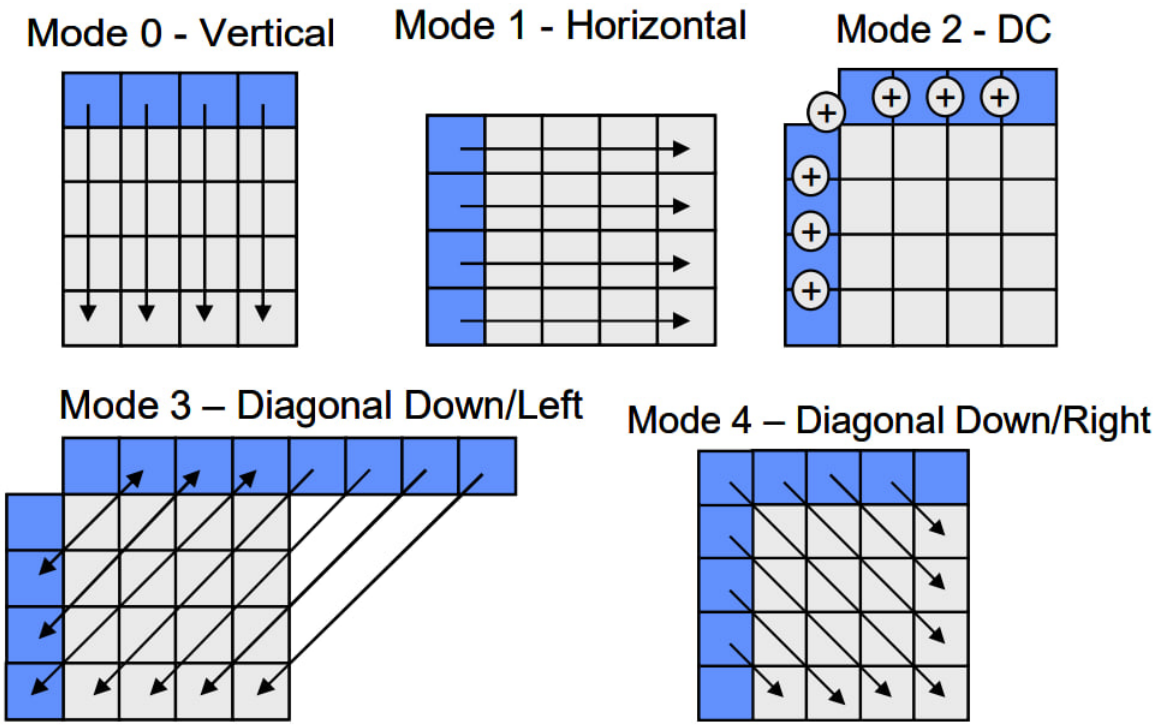

Рис.1.5. Режимы внутрикадрового предсказания H.264 [16]

- Поддержка аппаратного декодирования: Преимущества H.264 заключаются в расширенной поддержке аппаратного декодирования, которая снижает вычислительную нагрузку на устройства и повышает эффективность использования батареи мобильных устройств [16].

Недостатки H.264:

- Более высокая сложность кодирования: процесс кодирования может потребовать больше временных затрат на осуществление вычислений, особенно при высоком разрешении и скорости передачи данных;
- Ограниченная эффективность при очень высоком разрешении: H.264 очень эффективен для видео стандартной четкости, однако новые кодеки, такие как HEVC (H.265), обеспечивают более высокую эффективность при разрешении 4K и более высоких разрешениях [17].

1.5.2. Видеокодек VP8

VP8 - это кодек видео с открытым исходным кодом, поддерживаемый группой технологических компаний. Он был разработан с целью достижения высокой эффективности сжатия при низкой сложности декодирования [18]. VP8 базируется на концепции макроблоков, где каждый макроблок представляет собой изображение размером 16x16 пикселей яркости и двух блоков 8x8 пикселей цветности.

Особенности VP8:

- Референсные кадры - это кадры, которые используются для повышения эффективности кодирования. Они помогают предсказывать текущие кадры на основе сохранённых, что уменьшает объём данных, необходимых для передачи;
- Золотой кадр - копия предыдущего кадра, хранящая данные фона для восстановления скрытых областей при движении объектов;
- Субблоки - это мелкие части макроблока, которые кодируются независимо для большей гибкости при анализе движения. Режим SPLITMV позволяет каждому субблоку иметь свой собственный вектор движения, что эффективно для сложной последовательности кадров;
- Гибридное преобразование. Оно предназначено для эффективного кодирования остаточных частей [18];

Преимущества VP8:

- Низкая сложность декодирования позволяет использовать кодек на устройствах с низкой вычислительной способностью;
- VP8 достигает высокого качества при низких битрейтах, что особенно важно для потокового видео [18]. Однако для систем хранения видеоконтента с долгосрочным доступом плохая адаптивность может плохо сказаться на пользовательском опыте.

Недостатки VP8:

- VP8 уступает H.264 в эффективности сжатия динамично меняющихся кадров при одинаковом битрейте [18]. Это делает его менее подходящим для хранения видео с высокой детализацией и более подходящим для прямых трансляций;
- Поддержка аппаратного декодирования для VP8 меньше по сравнению с H.264 [18].

1.5.3. Сравнительный анализ технологий сжатия видеоконтента

Каждый из перечисленных ранее кодеков имеет ряд своих преимуществ и недостатков. Выбор того или иного кодека должен отталкиваться от эффективности решения, которое он предлагает для определённой проблемы или задачи. В этой работе рассматривается система с долгосрочным доступом к видеоконтенту, поэтому необходимо сделать выбор более подходящего кодека конкретно под эту задачу. Проведём сравнительный анализ ранее расмотренных видеокодеков:

- H.264 обеспечивает лучшее качество видео при битрейтах до 720p по сравнению с VP8 [19]. Это улучшает пользовательский опыт при использовании системы. Для среднего пользователя это качество как раз и является основным значением качества просмотра видео, так как видео с большим разрешением требуют большой пропускной способности сети и вычислительных мощностей устройства пользователя. Учитывая, что большинство контента на сегодняшний день потребляется именно с мобильных устройств, H.264 в данном случае имеет объективное преимущество;
- H.264 превосходит VP8 в скорости кодирования для большинства разрешений [19], что делает его предпочтительным для систем, где важна быстрая обработка видео. У VP8 скорость кодирования значительно ниже, что может замедлить процесс обработки;
- H.264 показывает более высокую точность в достижении целевого битрейта [19]. Это важно для систем с долгосрочным доступ к видеоконтенту, так как позволяет избежать непредсказуемого переполнения дискового пространства сервера. VP8 менее точен, что может привести к непредсказуемым результатам в плане размера файлов [19];
- H.264 поддерживает B-кадры, которые обеспечивают до 20% экономии битрейта за счет использования предсказания как от предыдущих, так и от будущих кадров. VP8 не поддерживает B-кадры, что ограничивает его способность к эффективному сжатию [19];
- H.264 поддерживается большинством устройств, браузерными окружениями и другим программным обеспечением. VP8 имеет ограниченную совместимость [19], что может привести к проблемам воспроизведения на определённых программных окружениях.

Кроме перечисленных выше параметров, также хочется уточнить, что H.264 принадлежит к семейству кодеков MPEG [16], поэтому он лучше совместим с технологиями, в которых MPEG-организация принимала участие при разработке. Это значит, что для стандарта MPEG-DASH кодек H.264 также является более предпочтительным. H.264.

В связи с этим, на текущий момент H.264 является оптимальным выбором для поставленной задачи благодаря его лучшей браузерной поддержке, совместимости с технологиями MPEG, высокой скорости кодирования и точности обработки битрейта. Это обеспечивает стабильное воспроизведение контента на большинстве устройств и окружений. Кроме того, этот кодек также больше ориентирован на

адаптивность, чем VP8, а также позволяет системам, которые его используют, лучше масштабироваться для работы с большими объемами данных. Кодек H.264 лучше всего подходит для системы с адаптивным стримингом видео и долгосрочным доступом к контенту.

1.6. Протоколы для передачи файлов между узлами сетей

При проектировании систем для загрузки и передачи файлов между узлами сетей возникает проблема выбора протокола их передачи. Эта задача актуальна и при работе с видео, так как видеоконтент отличается большим объемом и требует осознанного подхода к передаче. Существуют различные протоколы, которые предлагают разные возможности в зависимости от поставленных задач. Одни протоколы оптимизированы для передачи небольших сегментов данных, другие же лучше справляются с отправкой файлов целиком, обеспечивая их целостность и надежность доставки. Кроме того, важно также учитывать возможности поддержания безопасности системы при использовании того или иного протокола. Поэтому при выборе протокола для передачи данных необходимо отталкиваться от поставленных в проекте задач. В контексте рассматриваемой в ходе работы системы крайне важно поддерживать кроссплатформенность технологии на клиентской стороне, целостность и безопасность передачи контента.

1.6.1. Протокол HTTP

HTTP (HyperText Transfer Protocol) — это протокол передачи гипертекстовых документов, который является основным способом обмена данными в интернете [20]. HTTP был разработан для обмена текстовыми документами и ресурсами между веб-серверами и клиентами, однако со временем он стал основным механизмом для передачи данных любых типов, включая файлы или бинарные данные.

Основные принципы работы HTTP:

- HTTP представляет собой протокол прикладного уровня модели OSI (Open Systems Interconnection). Он работает в модели запрос-ответ, где клиент (обычно это веб-браузер) отправляет запрос к серверу, а сервер в ответ отправляет данные;
- Каждый HTTP запрос состоит из:
    - Метод запроса (GET, POST, PUT, DELETE и др.);

- URL (Uniform Resource Locator) — адрес ресурса;
- HTTP-версия (например, HTTP/1.1);
- Заголовки — дополнительные метаданные, которые могут содержать информацию о клиенте [20]. Заголовки можно использовать для передачи данных о текущей пользовательской сессии в зашифрованном виде, а также для разметки сегментов файлов при их загрузке в виде бинарных данных;
- Тело запроса — данные, передаваемые с запросом.

Пример взаимодействия между сервером и клиентом на основе протокола HTTP представлена на рисунке 1.6.

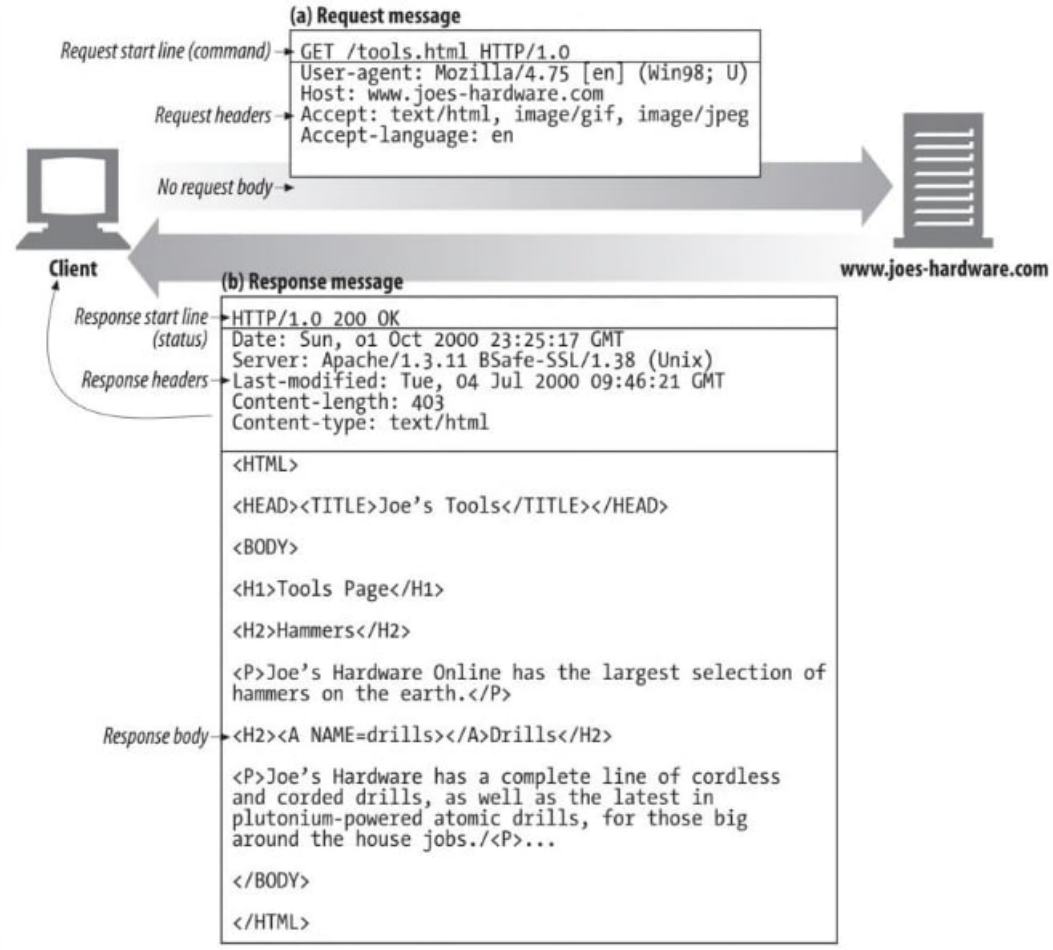

Рис.1.6. Пример взаимодействия между сервером и клиентов на основе протокола HTTP [21]

Рассмотрим различные способы передачи файл на сервер с помощью HTTP:

- Одним из самых распространенных способов отправки файлов через HTTP является использование заголовка "Content-Type"со значением "multipart/form-data"[20]. Он используется в основном при отправке данных

через HTML-формы, когда помимо файлов могут быть отправлены другие типы данных, такие как текстовые поля;

- Файлы можно передавать в виде бинарных данных. В этом способе файл передается в теле HTTP-запроса как непрерывная последовательность байтов. Как и в предыдущем способе, для передачи используется заголовок "Content-Type но со значением "application/octet-stream". Этот подход обычно используется для передачи больших файлов или когда необходимо передать двоичные данные, такие как изображения или видео. Преимущество данного способа неочевидно, однако он позволяет разбивать файл на клиентской стороне на несколько сегментов и отправлять его в виде нескольких частей. Разметку для этих сегментов можно передавать в заголовках запроса;
- Файлы могут быть закодированы в строку Base64 и отправлены в виде строки. Например, в теле запроса или в качестве части данных JSON. Такой способ не подходит для передачи больших файлов, так как требует дополнительных неэффективных вычислений на клиентской и серверной сторонах.

Преимущества использования протокола при загрузке файлов:

- Кроссплатформенность и поддержка всех браузеров позволяет передавать файлы с различных устройств и браузерных сред;
- HTTP позволяет отправлять файлы различными способами и предоставляет большую гибкость в формировании запросов к серверу благодаря наличию заголовков. Файлы можно предварительно разбить на сегменты на клиентской стороне и последовательно или параллельно отправлять на сервер;
- HTTP в совокупности с протоколом SSL/TLS позволяют обеспечить защищённое шифрованное соединение между клиентом сервером. Совокупность этих протоколов обобщённо имеет название HTTPS. HTTPS поддерживается по умолчанию всеми современными браузерами и не требует дополнительных настроек в отличие от других протоколов.

Недостатки использования протокола при загрузке файлов:

- Один из основных факторов, который может повлиять на отправку больших файлов — это размер HTTP-запроса. Спецификация HTTP не ограничивает размер запроса, однако серверы и браузеры могут накладывать ограничения на размер передаваемых данных [20]. В таком случае файл

необходимо предварительно обработать на стороне клиента и отправлять данные несколькими запросами в виде, например, сегментов;

- При отправке больших файлов HTTP-запросы могут сильно загружать сервер, так как загрузка файлов без предварительного преобразования на клиентской стороне требует их синхронной обработки на стороне сервера;
- Если передача файла занимает много времени, могут возникать проблемы с тайм-аутами на стороне клиента или сервера. Тайм-ауты могут возникать, если сервер или клиент не получают данных в течение установленного времени.

1.6.2. Протокол TFTP

TFTP (Trivial File Transfer Protocol) — это простой и легковесный протокол для передачи файлов по сети [22].

Протокол позволяет отправлять запросы на чтение и запись файлов:

- RRQ (Read Request) — запрос на чтение файла с сервера;
- WRQ (Write Request) — запрос на запись файла на сервер [22].

После запроса на чтение сервер начинает отправлять данные файла клиенту по блокам. Каждый блок данных сопровождается порядковым номером. После получения каждого блока клиент отправляет серверу ACK, что подтверждает успешную передачу. Для записи файла клиент отправляет блоки данных серверу, который подтверждает их получение. Если возникает ошибка, сервер отправляет сообщение об ошибке в ответ на запрос клиента. Когда файл полностью передан, клиент и сервер закрывают соединение [22].

Особенности и ограничения протокола TFTP:

- TFTP использует протокол UDP [22], он не гарантирует доставку пакетов и не имеет встроенных механизмов для повторной передачи данных;
- Протокол TFTP не включает механизмы аутентификации, авторизации или шифрования, что делает его менее безопасным;
- Стандартный размер пакета данных в TFTP — 512 байт [22], что ограничивает максимальный размер передаваемого файла. Для очень больших файлов, каждый из которых требует множества пакетов, это может существенно замедлять их обработку;
- Не поддерживается в современных браузерах.

1.6.3. Протокол FTP

FTP - это протокол, разработанный для надежной передачи файлов между клиентами и серверами в сетях с использованием TCP [23].

Основные цели и задачи протокола FTP:

- Перенос файлов: обеспечение механизма для передачи файлов между хостами, независимо от их операционных систем или файловых структур [23];
- Абстракция файловой структуры: протокол предоставляет общий интерфейс для работы с файлами [23], скрывая различия в файловых системах между клиентом и сервером;
- Поддержка взаимодействия человека с системой: FTP предлагает удобный способ работы для клиента, предоставляя команды для взаимодействия с сервером через текстовые запросы;

Особенности протокола:

- FTP использует два независимых соединения [23]:
    - Управляющее соединение (Control Connection): работает на 21 TCP-порту и используется для передачи команд и ответов между клиентом и сервером [23];
    - Соединение данных (Data Connection): работает на 20 TCP-порту и используется непосредственно для передачи файлов каталогов.
- Режимы передачи данных:
    - Активный режим (Active Mode): сервер открывает соединение для передачи данных на указанный клиентом адрес [23];
    - Пассивный режим (Passive Mode): сервер сообщает клиенту о доступном порте, и клиент инициирует соединение [23].

Основные команды FTP [23]:

- LIST: получение списка файлов и каталогов;
- RETR: загрузка файла с сервера;
- STOR: загрузка файла на сервер;
- DELE: удаление файла;
- CWD: изменение текущего каталога;
- QUIT: завершение сеанса.

Преимущества FTP:

- Протокол предоставляет команды для работы с файлами и каталогами: создание, удаление, переименование и перемещение;
- TCP, который использует FTP, гарантирует доставку всех пакетов [23];
- FTP поддерживает возобновление передачи файлов [23];
- Протокол обеспечивает универсальный подход для передачи как текстовых, так и двоичных данных.

Ограничения и недостатки FTP:

- Логины, пароли и данные передаются в незашифрованном виде [23]. Необходимо самостоятельно поддерживать шифрование на серверной и клиентской частях приложения;
- Протокол не поддерживает современные механизмы авторизации, такие как токены;
- Не поддерживается в современных браузерах.

1.6.4. Сравнительный анализ протоколов передачи файлов между узлами сетей

Ключевым требованием к протоколу для передачи файлов между клиентом и сервером является его кроссплатформенность. Протокол должен поддерживаться на всех браузерных окружениях и устройствах клиента. Из рассмотренных в данной главе протоколов этому требованию соответствует только протокол HTTP. Кроме того, вместе с этим протоколом во всех современных браузерах по умолчанию поддерживается также SSL/TLS [20], что обеспечивает безопасность передаваемых пользователем данных. Поэтому в качестве протокола для передачи данных между сервером и клиентом оптимальным выбором является HTTP.

В системах хранения и обработки видео зачастую используются разные серверы для выполнения этих функций в отдельности. Для передачи файлов между сервером обработки и сервером хранения также необходимо сделать выбор в пользу того или иного протокола. В данном случае нет необходимости беспокоиться о его поддержке, так как клиенты протокола можно установить на соответствующие сервера. Кроме того, сеть передачи между ними можно виртуализировать и скрыть от внешнего пользователя, на сегодняшний день с этим отлично справляются средства контейнеризации Docker, containerd, и другие. Это позволяет обеспечить безопасность между серверами, поэтому средства авторизации и аутентификации самого протокола играют посредственную роль. Таким образом, для межсерверного

обмена файлами подходит не только протокол HTTP, но и FTP, TFTP. Благодаря своему двухканальному режиму, более развёрнутому списку команд для работы с файлами и директориями, а также возобновлению сессий загрузки FTP является наиболее эффективным решением для межсерверного обмена файлами.

1.7. Выводы

В данной главе были рассмотрены технологии стриминга, сжатия и передачи видеоконтента для реализации серверной части системы. Проведён сравнительный анализ производительности, адаптивности и совместимости с современными браузерами соответствующих технологий.

По итогам сравнительного анализа были сделаны следующие выводы:

- Для стриминга видео целесообразно использовать протоколы MPEG–DASH и HLS, которые обеспечивают адаптивный стриминг, минимальные задержки и целостность данных при передаче. На серверной части системы необходимо поддерживать оба протокола, а на клиенте - в зависимости от браузерной среды: MPEG-DASH подходит для большинства браузеров, кроме Safari - для него более предпочтителен HLS;
- В качестве кодека для сжатия видео выбран H.264, так как он демонстрирует лучшее качество сжатия при меньших битрейтах, высокую производительность, поддержку аппаратного декодирования и совместимость с современными устройствами и протоколами;
- Для передачи данных между клиентом и сервером выбран HTTP, благодаря его кроссплатформенности, поддержке шифрования (с использованием SSL/TLS) и гибкости работы с сегментированной передачей данных. Для межсерверного обмена контентом был выбран FTP, который обеспечивает надёжную передачу файлов с возможностью возобновления загрузок.

1.8. Формулирование задачи и гипотезы её решения

Рассмотренные выше выводы формируют базу для определения цели и задач исследования, а также позволяют выдвинуть определённые гипотезы.

Цель исследования:

Создание приложения с системой адаптивного стриминга видео в нескольких синхронизированных потоках с возможностью навигации между ними, распре-

делённой сегментированной загрузкой контента и редактором синхронизации видеопотоков.

Для достижения цели были поставлены следующие задачи:

1. Разработка архитектуры системы распределённой сегментированной загрузки и раздачи видеоконтента;
2. Разработка головного сервиса для обработки пользовательских запросов, инициации сессий загрузки, распределения асинхронных задач между другими сервисами, формирования пользовательской выдачи и обновления в реальном времени статуса загрузки для клиента;
3. Реализация сервиса для распределённой обработки загрузки видеоконтента по сегментам и преобразования полученного видеоконтента в нужный кодек и в форматы различных протоколов адаптивного стриминга (DASH, HLS) для поддержания кроссплатформенности клиентов;
4. Разработка сервиса для хранения и раздачи видеоконтента;
5. Разработка вспомогательных сервисов для очистки прерванных сессий загрузки и удаления неиспользуемого контента (шедулеров);
6. Разработка архитектуры клиентских приложений для адаптивного стриминга видео и редактора синхронизации видеопотоков;
7. Разработка видеоплеера с поддержкой различных потоковых форматов в разных браузерах и синхронизации между видеопотоками;
8. Разработка браузерного редактора синхронизации видеопотоков;
9. Проведение нагрузочного и end-to-end тестирования и апробации реализованного приложения;
10. Разработка и настройка среды контейнеризации для автоматизированного развёртывания серверных и клиентских сервисов, СУБД, брокеров сообщений и балансировщиков нагрузки.

А также были сформулированы следующие гипотезы:

- Использование MPEG-DASH и HLS как основных протоколов стриминга обеспечит кроссплатформенность и адаптивность системы;
- MPEG-DASH предпочтителен для большинства браузеров благодаря поддержке переменного битрейта и минимизации задержек, тогда как HLS лучше всего подходит для экосистемы Apple, где он по умолчанию поддерживается;
- Применение кодека H.264 позволит достичь высокого качества видео при оптимальном размере файлов и минимальных накладных расходах на

обработку. Это обеспечит лучшую адаптивность, совместимость с протоколами (особенно MPEG-DASH) и поддержку большинства современных устройств;

- Использование HTTP как основного протокола передачи данных между клиентом и сервером гарантирует надёжность и безопасность (с использованием SSL/TLS), кроссплатформенность, а также гибкость при работе с сегментированной загрузкой контента. Это особенно актуально для систем с долгосрочным доступом к видеоконтенту;
- Использование FTP для межсерверного обмена файлами будет эффективным благодаря поддержке возобновляемых сессий, двухканального режима и развитых команд для работы с директориями;
- Выбор H.264 в качестве кодека сжатия видео позволит эффективно обрабатывать контент с высоким разрешением, минимизируя потери качества и обеспечивая широкий охват устройств и браузеров. Это ключевой фактор для систем с долгосрочным доступом к видеоконтенту;
- Разделение видео на сегменты и использование адаптивного стриминга (например, VBR) улучшит пользовательский опыт за счёт динамического выбора качества контента в зависимости от пропускной способности сети.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Apostolopoulos J.* Video streaming: Concepts, algorithms, and systems. — URL: https://www.researchgate.net/profile/John-Apostolopoulos/publication/228711326_Video_streaming_Concepts_algorithms_and_systems/links/551b57830cf2fdce84389bbe (дата обращения: 03.01.2025);

2 *Короткова Е. Н.* Мультимедийные средства массовой коммуникации: контент и технологии. // Известия Российского педагогического университета им. А. И. Герцена. — 2008. — С. 201—204;

3 *Cynthia J. Brame* Effective Educational Videos: Principles and Guidelines for Maximizing Student Learning from Video Content. // CBE–Life Sciences Education. — 2016;

4 *Ильина Е. Л., Латкин А. Н., Бочарова Э. А.* Видеомаркетинг как перспективное направление контент-маркетинга. // Символ науки. — 2017. — С. 102—104;

5 *Yani Xiao, Lan Wang, Ping Wang* Research on the Influence of Content Features of Short Video Marketing on Consumer purchase intentions. // Proceedings of the 2019 4th International Conference on Modern Management, Education Technology and Social Science — 2019;

6 *Mufti T., Mittal P., Gupta B.* A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services. — URL: https://eudl.eu/pdf/10.4108/eai.27-2-2020.2303255 (дата обращения: 16.01.2025);

7 *Мигратов Р.* ОБЛАЧНАЯ ПЛАТФОРМА ЯНДЕКСА И МЕТОД «КУЧЕВЫХ ОБЛАКОВ». // Студенческий вестник. — 2019. — № 43(93). — Часть 5. — С. 46—49;

8 Яндекс.Облако Документация — URL: https://yandex.cloud/ru/docs (дата обращения: 16.01.2025);

9 RTP: A Transport Protocol for Real-Time Applications. — URL: https://www.rfc-editor.org/rfc/rfc3550 (дата обращения: 03.01.2025);

10 MPEG-DASH. — URL: https://www.mpeg.org/standards/MPEG-DASH (дата обращения: 03.01.2025);

11 *Sodagar I.* The MPEG-DASH Standard for Multimedia Streaming Over the Internet. // IEEE MultiMedia. — 2011. — С. 62—67;

12 HTTP Live Streaming. — URL: https://www.rfc-editor.org/rfc/rfc8216 (дата обращения: 03.01.2025);

13 *Fecheyr-Lippens A.* A Review of HTTP Live Streaming. — URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf &doi=57d33cda30c2d497b694470aaa8b502613851fa5 (дата обращения: 03.01.2025);

14 *Bouzakaria N., Concolato C., Jean Le Feuvre* Overhead and performance of low latency live streaming using MPEG-DASH. // IEEE. — 2014;

15 *Richardson E.* Video Codec Design: Developing Image and Video Compression Systems. — The Robert Gordon University, Aberdeen, UK. — JOHN WILEY & SONS, LTD — 2004;

16 *Wiegand T., Sullivan Gary J., Bjontegaard G., Luthra A.* Overview of the H.264 / AVC Video Coding Standard. // IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY — 2003. — C. 560—576;

17 *Grois D., Marpe D., Mulayoff A., Itzhaky B., Hadar O.* Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders. // IEEE. — 2014;

18 *Bankoski J., Wilkins P., Xu Y.* Technical overview of VP8, an open source video codec for the web. // IEEE International Conference on Multimedia and Expo. — 2011;

19 *Sharrab Y. O., Sarhan, N. J.* Detailed Comparative Analysis of VP8 and H.264. // IEEE International Symposium on Multimedia/ — 2012;

20 Hypertext Transfer Protocol – HTTP/1.0. — URL: https://www.rfc-editor.org/rfc /rfc1945 (дата обращения: 03.01.2025);

21 *Gourley D., Totty B., Sayer M., Reddy S., Aggarwal A.* HTTP The Definition Guide. // O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol — 2002;

22 *Mohd Anuar Mat Isa, Nur Nabila Mohamed, Hashim H., Syed Farid Syed Adnan, Jamalul-lail Ab Manan, Mahmod R.* A lightweight and secure TFTP protocol for smart environment. // International Symposium on Computer Applications and Industrial Electronics (ISCAIE). — 2012;

23 FILE TRANSFER PROTOCOL (FTP). — URL: https://www.rfc-editor.org/rfc /rfc959 (дата обращения: 03.01.2025).