|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 | Робототехника и комплексная автоматизация (РК) |
| КАФЕДРА | Системы автоматизированного проектирования (РК6) |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***«Приложение для видеосвязи на Golang»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-76Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Онюшев А.А.** |
|  | (Подпись, дата) | И.О. Фамилия |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

*2024 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 РК6

А.П. Карпенко

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Приложение для видеосвязи на Golang \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_РК6-11М\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Онюшев Артем Андреевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.) \_учебная\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Техническое задание:\_1) Изучить аналоги приложений\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) Изучить возможные технологии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) Рассмотреть варианты применения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4) Спроектировать возможную реализацию приложения .\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5) Придумать возможные оптимизации и улучшения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на 17 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «8» октября 2024 г.

|  |  |  |
| --- | --- | --- |
| **Руководитель НИР** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |
| **Студент** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Онюшев А.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафеде

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc190385308)

[1. Анализ современных решений для видеосвязи 5](#_Toc190385309)

[2. Выбор технологий и инструментов для разработки 8](#_Toc190385310)

[3. Проектирование архитектуры приложения 12](#_Toc190385311)

[4. Реализация приложения для видеосвязи 16](#_Toc190385312)

[5. Оптимизация и производительность 20](#_Toc190385313)

[ЗАКЛЮЧЕНИЕ 22](#_Toc190385314)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc190385315)

ВВЕДЕНИЕ

В современном мире видеосвязь стала неотъемлемой частью как повседневной жизни, так и профессиональной деятельности. С развитием технологий и увеличением спроса на удаленное взаимодействие возрастает потребность в надежных, производительных и безопасных решениях для видеоконференций и видеозвонков. Однако многие существующие платформы имеют ограничения, которые предстоит импортозаместить. В связи с этим разработка специализированного приложения для видеосвязи с использованием современных технологий, таких как язык программирования Golang, представляет собой актуальную задачу.

Целью данной работы является исследование и проектирование приложения для видеосвязи на языке Golang, обеспечивающего высокую производительность, низкие задержки и безопасность передачи данных.

Основные задачи исследования:

* Провести анализ существующих решений для видеосвязи и выявить их преимущества и недостатки.
* Изучить возможности языка Golang и его экосистемы для разработки приложений реального времени.
* Разработать архитектуру приложения, включая клиентскую и серверную части.
* Рассмотреть передачу видеопотоков с использованием современных протоколов, таких как WebRTC.
* Обеспечить разработанное приложение высокой производительностью.

Акцентирование внимания на данной проблеме не только позволяет создать удобное приложение для видеосвязи, но также может привести к разработке более эффективных и независимых систем, нежели иностранные аналоги.

1. Анализ современных решений для видеосвязи

**1.1. Обзор существующих приложений для видеосвязи**

* *Zoom*: Одно из самых известных приложений для видеоконференций. Zoom предлагает высокое качество видео и аудио, поддержку большого количества участников, а также интеграцию с календарями и инструментами для совместной работы. Однако приложение критикуют за проблемы с безопасностью и зависимость от облачной инфраструктуры, а также платную подписку. Является иностранным ПО.



Рис.1 Логотип Zoom

* *Microsoft Teams*: Решение, интегрированное в экосистему Microsoft 365. Teams поддерживает видеозвонки, чаты, совместную работу над документами и интеграцию с другими сервисами Microsoft. Основным недостатком является высокая ресурсоемкость приложения. Является иностранным ПО.



Рис.2 Логотип Microsoft Teams

* *Skype*: Одно из первых приложений для видеосвязи, которое до сих пор используется благодаря своей простоте и надежности. Однако Skype уступает современным решениям в плане функциональности и качества видеопотоков, выглядит устаревшим. Является иностранным ПО.



Рис.3 Логотип Skype

* *Discord*: Изначально разработанный для геймеров, Discord стал популярным инструментом для видеосвязи благодаря низким задержкам и поддержке сообществ. Однако он не предназначен для корпоративного использования и запрещен на территории РФ.



Рис.4 Логотип Skype

**1.2. Требования к современным приложениям для видеосвязи**

На основе анализа существующих решений можно выделить ключевые требования к современным приложениям для видеосвязи:

1. **Производительность**:
   * Низкие задержки при передаче аудио и видео.
   * Поддержка высокого качества видеопотоков (HD, 4K).
   * Оптимизация для работы на устройствах с ограниченными ресурсами.
2. **Масштабируемость**:
   * Возможность поддержки большого количества участников.
   * Работа в распределенных сетях с минимальной нагрузкой на серверы.
3. **Безопасность**:
   * Шифрование данных (end-to-end encryption).
   * Защита от атак, таких как DDoS и MITM.
   * Аутентификация и авторизация пользователей.
4. **Кроссплатформенность**:
   * Поддержка различных операционных систем (Windows, macOS, Linux, iOS, Android).
   * Интеграция с браузерами через WebRTC.
5. **Простота использования**:
   * Интуитивно понятный интерфейс.
   * Минимальные требования к настройке со стороны пользователя.
6. **Гибкость и кастомизация**:
   * Возможность интеграции с другими сервисами (календари, CRM, инструменты для совместной работы).
   * Поддержка различных кодеков и форматов видео.

**Итог**

Анализ современных решений для видеосвязи показывает, что, несмотря на наличие множества приложений, каждое из них имеет свои ограничения. Это создает возможность для разработки нового решения, которое объединит в себе высокую производительность, безопасность и простоту использования. Использование языка Golang, благодаря его производительности и простоте, может стать ключевым фактором в создании такого приложения.

1. Выбор технологий и инструментов для разработки

**2.1. Язык программирования Golang: особенности и преимущества**

Язык программирования Golang (Go) был разработан компанией Google в 2009 году и с тех пор завоевал популярность благодаря своей простоте, производительности и эффективности. Для разработки приложения для видеосвязи Golang предлагает следующие преимущества:

* **Высокая производительность**: Golang компилируется в машинный код, что обеспечивает высокую скорость выполнения программ. Это особенно важно для приложений реального времени, таких как видеосвязь.
* **Простота и читаемость кода**: Синтаксис Golang минималистичен и легко читаем, что упрощает разработку и поддержку кода.
* **Поддержка многопоточности**: Golang имеет встроенную поддержку горутин (goroutines) и каналов (channels), что позволяет эффективно работать с параллельными задачами, такими как обработка видеопотоков.
* **Кроссплатформенность**: Golang поддерживает компиляцию под различные операционные системы (Windows, macOS, Linux) и архитектуры, что делает его идеальным для создания кроссплатформенных приложений.
* **Богатая экосистема**: Golang имеет множество библиотек и инструментов для работы с сетью, шифрованием, мультимедиа и другими задачами, что ускоряет процесс разработки.

**2.2. Библиотеки и фреймворки для работы с видеопотоками**

Для реализации функциональности видеосвязи в Golang можно использовать следующие библиотеки и инструменты:

* *Pion WebRTC*: Библиотека для реализации протокола WebRTC на Golang. WebRTC является стандартом для передачи аудио и видео в реальном времени через браузеры и мобильные приложения. Pion WebRTC поддерживает создание как клиентской, так и серверной части приложения.
* *GStreamer*: Мощный фреймворк для обработки мультимедиа, который может быть интегрирован с Golang через языковые привязки. GStreamer поддерживает множество кодеков и форматов, что делает его универсальным инструментом для работы с видеопотоками.
* *FFmpeg*: Популярная библиотека для обработки аудио и видео. Хотя FFmpeg написан на C, его можно использовать в Golang через обертки (bindings).
* *gRPC*: Фреймворк для создания высокопроизводительных RPC-сервисов. gRPC может быть использован для передачи данных между клиентом и сервером, особенно в распределенных системах.

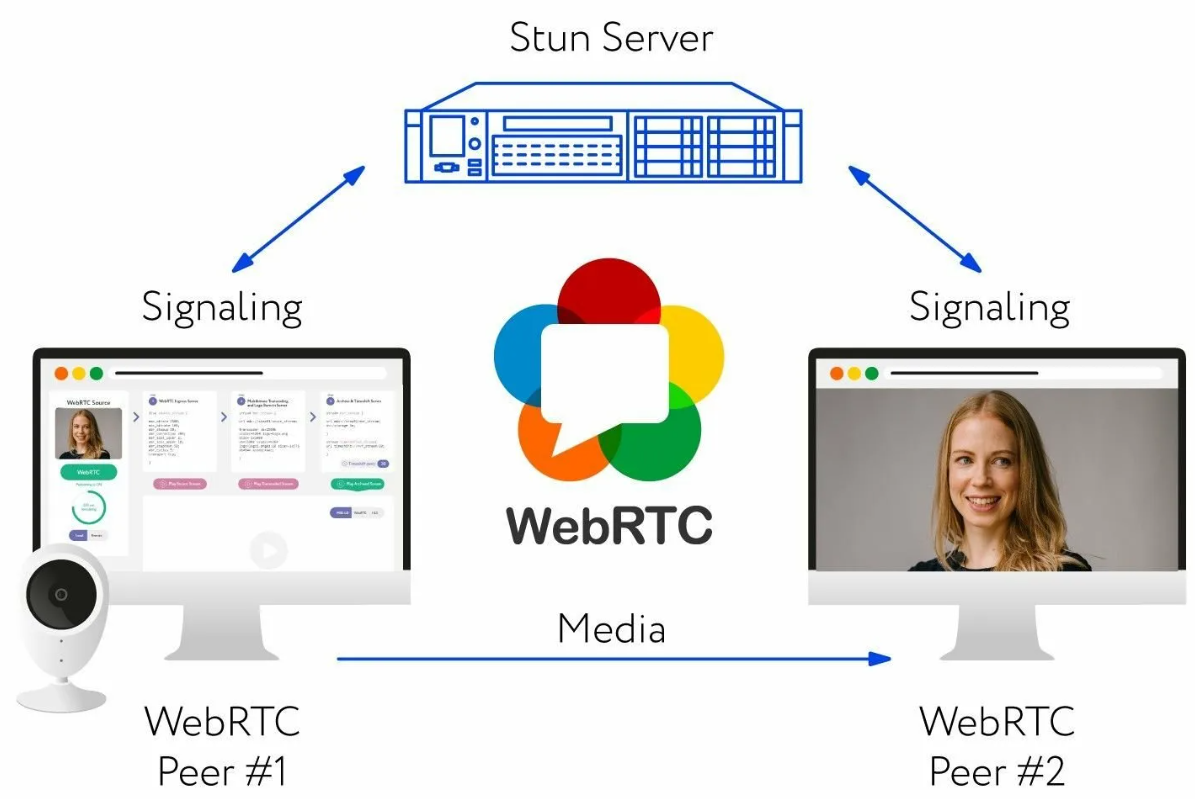


Рис. 5 Наглядный пример устройства WebRTC

**2.3. Протоколы передачи данных**

Для передачи аудио и видео в реальном времени используются различные протоколы, каждый из которых имеет свои особенности:

* **WebRTC (Web Real-Time Communication)**:
  + *Описание*: Открытый стандарт для передачи аудио, видео и данных между браузерами и приложениями.
  + *Преимущества*: Низкие задержки, поддержка P2P-соединений, шифрование данных.
  + *Недостатки*: Требует использования сигнальных серверов (signaling servers) и серверов TURN/STUN для обхода NAT.
* **SIP (Session Initiation Protocol)**:
  + *Описание*: Протокол для установления, управления и завершения мультимедийных сессий.
  + *Преимущества*: Широко используется в VoIP-системах, поддерживает интеграцию с корпоративными решениями.
  + *Недостатки*: Сложность настройки и интеграции, требует дополнительных протоколов для передачи медиа (например, RTP).
* **RTP (Real-time Transport Protocol)**:
  + *Описание*: Протокол для передачи аудио и видео в реальном времени.
  + *Преимущества*: Оптимизирован для работы с мультимедиа, поддерживает множество кодеков.
  + *Недостатки*: Требует использования дополнительных протоколов для управления сессиями (например, SIP).

Для разработки приложения на Golang наиболее предпочтительным является использование WebRTC, так как он обеспечивает низкие задержки и поддерживает P2P-соединения, что снижает нагрузку на серверы.

**2.4. Инструменты для обработки аудио и видео**

Для обработки аудио и видеопотоков в приложении могут быть использованы следующие инструменты:

* **Кодеки**:
  + *Видео*: H.264, VP8, VP9.
  + *Аудио*: Opus, AAC.  
    Эти кодеки обеспечивают высокое качество при минимальной нагрузке на сеть.
* **Библиотеки для обработки медиа**:
  + *libavcodec* (часть FFmpeg): Поддержка множества кодеков и форматов.
  + *Pion WebRTC*: Встроенная поддержка кодеков, используемых в WebRTC.
* **Инструменты для анализа и отладки**:
  + *Wireshark*: Для анализа сетевого трафика и отладки протоколов.
  + *GStreamer tools*: Для тестирования и отладки медиапотоков.

**Итог**

Выбор технологий и инструментов для разработки приложения для видеосвязи на Golang основывается на требованиях к производительности, безопасности и простоте интеграции. Golang, благодаря своей производительности и поддержке многопоточности, является идеальным выбором для реализации таких задач. Использование WebRTC и библиотек, таких как Pion WebRTC, позволяет создать современное и эффективное решение для видеосвязи.

1. Проектирование архитектуры приложения

**3.1. Модульная структура приложения**

Для обеспечения гибкости и удобства поддержки приложение должно быть разделено на несколько модулей, каждый из которых отвечает за определенную функциональность:

1. **Клиентский модуль**:
   * Отвечает за взаимодействие с пользователем (интерфейс, управление звонками).
   * Обрабатывает ввод и вывод аудио/видео данных.
   * Управляет подключением к серверу и другими клиентами.
2. **Серверный модуль**:
   * Обрабатывает запросы от клиентов (регистрация, аутентификация, управление сессиями).
   * Координирует передачу данных между участниками видеозвонка.
   * Обеспечивает работу сигнальных серверов (signaling) для установления P2P-соединений.
3. **Модуль обработки медиа**:
   * Отвечает за кодирование и декодирование аудио/видео потоков.
   * Обрабатывает данные с использованием выбранных кодеков (например, H.264, Opus).
   * Управляет передачей данных через протоколы WebRTC или RTP.
4. **Модуль безопасности**:
   * Обеспечивает шифрование данных (например, с использованием DTLS и SRTP).
   * Управляет аутентификацией и авторизацией пользователей.
   * Защищает систему от атак (например, DDoS, MITM).
5. **Модуль логирования и мониторинга**:
   * Собирает данные о работе приложения (ошибки, задержки, использование ресурсов).
   * Предоставляет инструменты для анализа производительности и устранения неполадок.

**3.2. Взаимодействие клиента и сервера**

Взаимодействие между клиентом и сервером в приложении для видеосвязи строится на основе следующих компонентов:

1. **Сигнальный сервер (Signaling Server)**:
   * Используется для обмена метаданными между клиентами (например, IP-адреса, информация о кодеках).
   * Реализуется с использованием протоколов WebSocket или HTTP/2.
   * Пример реализации на Golang: использование библиотеки *gorilla/websocket*.
2. **P2P-соединения (Peer-to-Peer)**:
   * После обмена метаданными клиенты устанавливают прямое соединение через WebRTC.
   * Для обхода NAT и файрволов используются серверы TURN и STUN.
3. **Серверная координация**:
   * В случае групповых видеозвонков сервер может выступать в роли медиа-микшера, объединяя потоки от нескольких участников и передавая их каждому клиенту.



Рис. 6 Устройство Peer-to-peer соединения

**3.3. Обработка и передача видеопотоков**

Обработка видеопотоков включает следующие этапы:

1. **Захват видео и аудио**:
   * Использование библиотек для захвата данных с камеры и микрофона (например, *Pion WebRTC* или *GStreamer*).
2. **Кодирование и декодирование**:
   * Применение кодеков для сжатия данных (например, H.264 для видео, Opus для аудио).
   * Использование библиотек, таких как *libavcodec* (FFmpeg) или встроенных возможностей WebRTC.
3. **Передача данных**:
   * Использование протокола WebRTC для передачи аудио и видео в реальном времени.
   * Обеспечение минимальных задержек и высокой пропускной способности.
4. **Адаптация к качеству сети**:
   * Реализация механизмов адаптации качества видеопотока в зависимости от пропускной способности сети (например, использование *Adaptive Bitrate Streaming*).

**3.4. Обеспечение безопасности и шифрования данных**

Безопасность является критически важным аспектом приложения для видеосвязи. Основные меры безопасности включают:

1. **Шифрование данных**:
   * Использование протоколов DTLS (Datagram Transport Layer Security) и SRTP (Secure Real-time Transport Protocol) для шифрования аудио и видеопотоков.
   * Обеспечение end-to-end шифрования для защиты данных от перехвата.
2. **Аутентификация и авторизация**:
   * Реализация механизмов аутентификации пользователей (например, OAuth, JWT).
   * Управление доступом к видеозвонкам (например, пароли для комнат, ограничение доступа по ролям).
3. **Защита от атак**:
   * Использование механизмов для предотвращения DDoS-атак (например, ограничение запросов, использование CDN).
   * Реализация защиты от атак MITM (Man-in-the-Middle) с использованием сертификатов и шифрования.
4. **Аудит и мониторинг**:
   * Ведение журналов событий для отслеживания подозрительной активности.
   * Регулярное тестирование системы на уязвимости.

**Итог**

Проектирование архитектуры приложения для видеосвязи на Golang требует тщательного подхода к организации модулей, взаимодействию клиента и сервера, обработке видеопотоков и обеспечению безопасности. Использование современных технологий, таких как WebRTC, и библиотек, таких как Pion WebRTC, позволяет создать производительное и надежное решение.

1. Реализация приложения для видеосвязи

Реализация приложения для видеосвязи на Golang включает несколько этапов: создание клиентской и серверной частей, интеграцию с протоколами передачи данных, а также тестирование и отладку. В этом разделе подробно рассмотрим ключевые аспекты реализации.

**4.1. Разработка клиентской части**

Клиентская часть приложения отвечает за взаимодействие с пользователем и обработку аудио/видео данных. Основные этапы разработки:

1. **Интерфейс пользователя**:
   * Создание простого и интуитивно понятного интерфейса для управления видеозвонками.
   * Использование библиотек для создания графического интерфейса, таких как *Fyne* или *web-интерфейс* на основе HTML/JavaScript.
2. **Захват аудио и видео**:
   * Использование библиотеки *Pion WebRTC* для захвата данных с камеры и микрофона.
3. **Установление соединения**:
   * Использование WebSocket для обмена сигнальными данными с сервером.
4. **Обработка входящих и исходящих потоков**:
   * Настройка обработчиков для входящих аудио/видео потоков.

**4.2. Разработка серверной части**

Серверная часть приложения отвечает за управление соединениями, обработку сигнальных данных и координацию видеозвонков. Основные этапы разработки:

1. **Сигнальный сервер**:
   * Реализация сервера на основе WebSocket для обмена метаданными между клиентами.
2. **Управление сессиями**:
   * Создание системы для управления видеозвонками (создание комнат, добавление участников).
3. **Интеграция с TURN/STUN серверами**:
   * Настройка серверов для обхода NAT и файрволов.

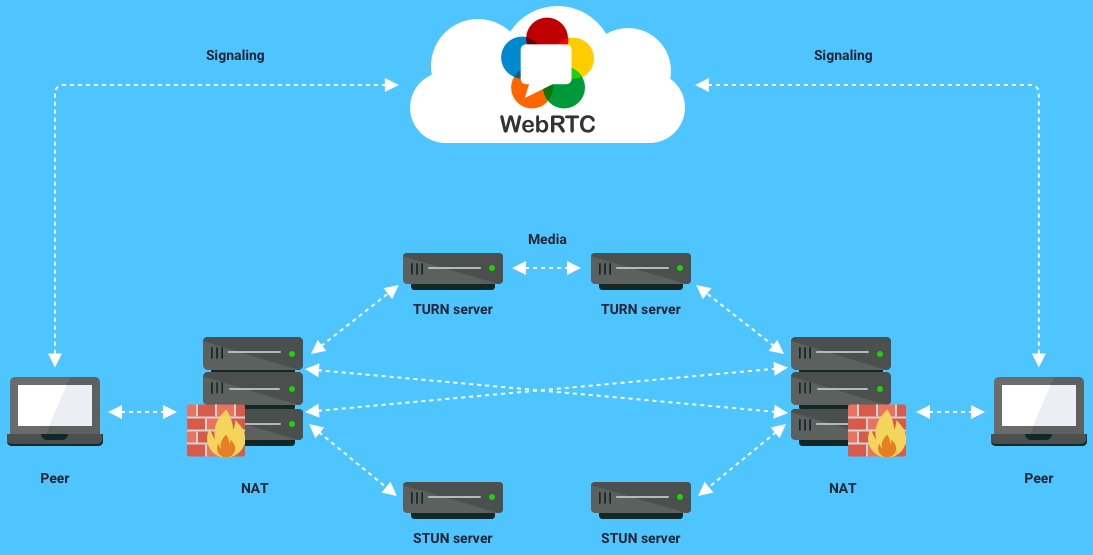


Рис. 7 Схематичное изображение работы сервера.

**4.3. Интеграция с протоколами передачи данных**

Для передачи аудио и видео данных используются протоколы WebRTC и RTP. Основные этапы интеграции:

1. **Настройка WebRTC**:
   * Создание и настройка объектов *PeerConnection* для управления соединениями.
2. **Передача данных через RTP**:
   * Использование библиотек для работы с RTP-пакетами.

**4.4. Тестирование и отладка приложения**

Тестирование и отладка являются важными этапами разработки. Основные шаги:

1. **Модульное тестирование**:
   * Написание тестов для отдельных компонентов приложения (например, обработка видеопотоков).
   * Использование встроенного фреймворка для тестирования в Golang.
2. **Интеграционное тестирование**:
   * Тестирование взаимодействия между клиентом и сервером.
   * Использование инструментов, таких как *Postman* для тестирования API.
3. **Тестирование под нагрузкой**:
   * Проверка производительности приложения при большом количестве участников.
   * Использование инструментов, таких как *Apache JMeter*.
4. **Отладка**:
   * Использование встроенных инструментов отладки в IDE (например, *Visual Studio Code*).
   * Логирование ключевых событий для анализа проблем.
5. Оптимизация и производительность

**5.1. Методы оптимизации видеопотоков**

Для обеспечения плавной передачи видеопотоков и минимизации нагрузки на сеть используются следующие методы оптимизации:

1. **Сжатие данных**:
   * Использование современных кодеков, таких как H.264, VP8 или VP9, которые обеспечивают высокое качество при минимальном размере данных.
2. **Адаптивное изменение битрейта (Adaptive Bitrate Streaming, ABS)**:
   * Автоматическое изменение качества видеопотока в зависимости от пропускной способности сети.
   * Реализация ABS с использованием библиотек, таких как *Pion WebRTC* или *GStreamer*.
3. **Буферизация и контроль задержек**:
   * Использование буферов для сглаживания задержек и потерь пакетов.
   * Настройка размера буферов и алгоритмов их управления.
4. **Оптимизация использования ресурсов**:
   * Уменьшение нагрузки на процессор и память за счет оптимизации кода и использования аппаратного ускорения (например, GPU).

**5.2. Снижение задержек и улучшение качества связи**

Задержки являются одной из основных проблем в приложениях для видеосвязи. Для их минимизации используются следующие подходы:

1. **Оптимизация сетевых протоколов**:
   * Использование протокола WebRTC, который обеспечивает низкие задержки за счет P2P-соединений.
   * Настройка TURN/STUN серверов для минимизации времени установления соединения.
2. **Приоритизация трафика**:
   * Использование QoS (Quality of Service) для приоритизации аудио и видеопотоков.
   * Пример настройки QoS в сетевом оборудовании.
3. **Оптимизация кодека**:
   * Выбор кодеков с низкой задержкой, таких как Opus для аудио и VP8 для видео.
   * Настройка параметров кодирования для минимизации задержек.
4. **Локальная обработка данных**:
   * Обработка аудио и видео данных на стороне клиента для уменьшения нагрузки на сервер.

**Итог**

Оптимизация и обеспечение высокой производительности приложения для видеосвязи требуют комплексного подхода, включающего оптимизацию видеопотоков, снижение задержек и тестирование под нагрузкой. Использование современных технологий и инструментов позволяет создать приложение, которое обеспечивает высокое качество связи даже в сложных условиях.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной научно-исследовательской работы была разработана архитектура приложения для видеосвязи на языке программирования Golang. Основной целью работы являлось создание высокопроизводительного, масштабируемого и безопасного решения, которое могло бы конкурировать с существующими платформами для видеоконференций. В процессе исследования были решены следующие задачи:

1. Проведен анализ современных решений для видеосвязи, выявлены их преимущества и недостатки, а также сформулированы требования к разрабатываемому приложению.
2. Выбраны технологии и инструменты для разработки, включая язык программирования Golang, библиотеку Pion WebRTC и протоколы передачи данных (WebRTC, RTP).
3. Разработана архитектура приложения, включающая модульную структуру, взаимодействие клиента и сервера, обработку видеопотоков и обеспечение безопасности.
4. Проведены примеры возможной оптимизации приложения для обеспечения высокой производительности, включая снижение задержек, адаптацию к качеству сети и тестирование под нагрузкой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Golang Tutorials [Электронный ресурс] ‒ URL: <https://go.dev/tour/welcome/1> (дата обращения: 09.10.2024 - 24.10.2024)
2. Официальная документация Golang [Электронный ресурс] ‒ URL: <https://golang.org/doc/> (дата обращения: 17.10.2024)
3. Pion WebRTC: библиотека для работы с WebRTC на Golang [Электронный ресурс] – URL: <https://github.com/pion/webrtc> (дата обращения: 20.10.2024 - 30.10.2024)
4. GStreamer: фреймворк для обработки мультимедиа [Электронный ресурс] – URL: <https://gstreamer.freedesktop.org/> (дата обращения: 20.10.2024 - 30.10.2024)
5. RTP - протокол для передачи мультимедиа [Электронный ресурс] – URL: <https://tools.ietf.org/html/rfc3550> (дата обращения: 01.11.2024)
6. DTLS - протокол для шифрования данных [Электронный ресурс] – URL:  <https://tools.ietf.org/html/rfc6347>  (дата обращения: 14.11.2024)
7. SRTP - протокол для шифрования данных [Электронный ресурс] – URL:  <https://tools.ietf.org/html/rfc3711>  (дата обращения: 15.11.2024)
8. Документация по адаптивному битрейту (Adaptive Bitrate Streaming) [Электронный ресурс] – URL: <https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=153508> (дата обращения: 21.11.2024)
9. Официальная документация по WebSocket в Golang [Электронный ресурс] – URL: <https://pkg.go.dev/github.com/gorilla/websocket> (дата обращения: 24.11.2024)