



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, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

*К КУРСОВОЙ РАБОТЕ*

*НА ТЕМУ:*

*«Протокол бессерверных конференций»*

Студент ИУ7-31М  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

Клименко А. К.  
(И. О. Фамилия)

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата)

Никульшин А. М.  
(И. О. Фамилия)

*2025 г.*

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>4</b>
<b>1 Аналитический раздел</b>	<b>5</b>
1.1 Актуальность . . . . .	5
1.2 Поточковая передача данных в реальном времени . . . . .	5
1.3 Распределение вычислений . . . . .	6
1.4 Микширование . . . . .	6
1.5 Устойчивость конференции . . . . .	7
1.6 Защита информации . . . . .	7
<b>2 Конструкторский раздел</b>	<b>9</b>
2.1 Функциональные требования к протоколу . . . . .	9
2.2 Роли участников конференции . . . . .	9
2.3 Типы сообщений . . . . .	9
2.4 Медиа форматы . . . . .	10
2.5 Взаимодействия сторон . . . . .	10
2.6 Шифрование . . . . .	12
2.7 Переподключение . . . . .	14
2.8 Определение качества каналов связи . . . . .	14
<b>3 Технологический раздел</b>	<b>15</b>
3.1 Средства разработки . . . . .	15
3.2 Листинги процедуры приглашения . . . . .	15
3.3 Сборка и запуск . . . . .	18
3.4 Тестирование ПО . . . . .	20
3.5 Трасировка пакетов . . . . .	20
<b>4 Исследовательский раздел</b>	<b>23</b>
4.1 Постановка исследования . . . . .	23
4.2 Результаты . . . . .	23
<b>ЗАКЛЮЧЕНИЕ</b>	<b>24</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>25</b>

# ВВЕДЕНИЕ

Целью данной работы является проектирование протокола проведения бессерверных аудио видео конференций. Для достижения поставленной цели необходимо решить следующие задачи:

- изучить схожие протоколы;
- спроектировать протокол бессерверных конференций;
- реализовать ПО осуществляющее взаимодействие по разработанному протоколу;
- протестировать ПО.

# 1 Аналитический раздел

## 1.1 Актуальность

Одной из наиболее актуальных задач в современном мире является обеспечение эффективного взаимодействия между удаленными пользователями. Бессерверные аудио-видео конференции представляют собой перспективное решение, позволяющее организовывать виртуальные встречи без необходимости использования централизованных серверов.

Традиционные подходы к организации конференц-связи, основанные на клиент-серверной архитектуре, сталкиваются с рядом ограничений. Во-первых, они требуют наличия мощной серверной инфраструктуры, которая должна обрабатывать весь трафик участников, что приводит к высоким эксплуатационным расходам. Во-вторых, централизованные системы являются уязвимыми к сбоям и атакам, так как отказ сервера может привести к полной потере связи между пользователями. В-третьих, такие решения зачастую ограничены в масштабируемости, что затрудняет их применение в крупномасштабных распределенных средах.

Бессерверные аудио-видео конференции, в свою очередь, обладают рядом преимуществ. Они позволяют организовывать виртуальные встречи напрямую между участниками, без необходимости использования централизованных серверов. Это повышает отказоустойчивость системы, поскольку отказ одного из клиентов не приводит к полной потере связи. Кроме того, данный подход способствует снижению эксплуатационных расходов и упрощает процесс масштабирования.

Разработка протокола для организации бессерверных аудио-видео конференций является важной научно-технической задачей, имеющей высокую практическую значимость. Результаты исследования могут найти применение в различных областях, таких как дистанционное обучение, удаленная работа, проведение виртуальных совещаний и конференций, а также во многих других сценариях, требующих эффективного взаимодействия между удаленными участниками.

## 1.2 Потокковая передача данных в реальном времени

Говоря о аудио-видео конференциях, следует выделить две основные составляющие потоков данных – аудио поток и видео поток. Аудио поток, как правило, имеет более высокий приоритет по сравнению с видео потоком, поскольку бесперебойная передача звука является критически важной для обеспечения качественного взаимодействия участников.

Основными протоколами транспортного уровня, используемыми при передаче данных в режиме реального времени, являются TCP, UDP. На прикладном уровне часто используются такие протоколы как RTP (Real-Time Transport Protocol [1]), RTCP (Real-Time Transport Control Protocol [1]), SIP (Session Initiation Protocol [2]), H.323 [3].

## 1.3 Распределение вычислений

При централизованном проведении конференций основная нагрузка ложится на сервер, в то время как при проведении бессерверных конференций она распределяется по клиентам. Для обеспечения наилучшего QoS необходимо учитывать вычислительные возможности каждого клиента, а также пропускную способность каналов, объединяющих их.

В бессерверной архитектуре каждый участник конференции выступает в качестве автономного узла, ответственного за обработку и передачу мультимедийного контента. Это требует от клиентских устройств достаточной вычислительной мощности для кодирования, декодирования и синхронизации аудио, видео и данных в режиме реального времени.

$$Perf_{client} = f(CPU, GPU, RAM) \quad (1.1)$$

Пропускная способность каналов связи также играет критическую роль, определяя максимально возможное качество и разрешение передаваемых потоков. Недостаточная пропускная способность может привести к снижению качества и стабильности соединения, а также к возникновению задержек и потерь пакетов.

$$Bandwidth_{channel} = f(network\ speed, distance, noises) \quad (1.2)$$

Для достижения оптимального QoS в бессерверных конференциях необходимо динамически адаптировать параметры кодирования и передачи мультимедийных потоков к возможностям каждого участника. Это может быть реализовано с помощью механизмов согласования характеристик сессии и адаптивного управления скоростью передачи данных.

## 1.4 Микширование

При большом числе участников в бессерверных видеоконференциях имеет смысл применять микширование передаваемых видео потоков. Это позволяет уменьшить общий объем передаваемых данных, снизив нагрузку на каналы связи, но при этом требует значительных вычислительных ресурсов от клиентских устройств.

Процесс микширования видео можно представить в виде следующих основных шагов:

1. Получение видео кадров от соседних узлов.
2. Декодирование полученных кадров.
3. Композиция единого кадра путем пространственного объединения (склеивания) исходных кадров.
4. Кодирование скомпонованного кадра.
5. Передача закодированного кадра всем соседним узлам.

При этом, на этапе формирования общего кадра необходимо учитывать, нужен ли тот или иной видео поток каждому из соседних участников. Это позволяет оптимизировать состав композитного кадра и снизить вычислительную нагрузку.

Микширование аудио потоков может быть реализовано схожим образом, с формированием отдельных аудио каналов для каждого участника конференции. Это требует дополнительных вычислений, связанных с микшированием, кодированием и передачей аудио данных.

Эффективность микширования определяется соотношением выигрыша в объеме передаваемых данных и затратами вычислительных ресурсов на клиентских устройствах. Для поддержания высокого качества и низкой задержки передачи мультимедийного контента необходим тщательный подбор алгоритмов микширования и кодирования с учетом возможностей участников конференции.

## 1.5 Устойчивость конференции

Отключение ключевого участника конференции, через которого проходит большое число потоков данных от других участников, является одной из проблем распределенных вычислений. Решением этой проблемы может послужить механизм, обеспечивающий поддержание «запасных» каналов передачи между клиентами.

## 1.6 Защита информации

Для защиты конференции необходимо использовать защищенные каналы передачи. На каждое соединение инициализируется собственная сессия, вместо того, чтобы использовать одну сессию на всех участников конференции.

Плюсы такого подхода:

- Повышение безопасности за счет использования индивидуальных криптографических ключей для каждого соединения.
- Улучшение масштабируемости системы, так как каждое соединение обрабатывается независимо.
- Более гибкое управление доступом и привилегиями участников конференции.

Минусы такого подхода:

- Увеличение вычислительной нагрузки на клиентские устройства, связанное с необходимостью поддержания множества криптографических сессий.
- Более высокие требования к пропускной способности канала связи, обусловленные необходимостью передачи дополнительных данных, связанных с криптографической защитой.

В программной реализации протокола применяется библиотека OpenSSL [4] для осуществления защищенных соединений. Использование OpenSSL позволяет обеспечить совместимость с широким спектром криптографических алгоритмов, а также реализовать такие механизмы безопасности, как аутентификация, шифрование и целостность передаваемых

данных. Кроме того, OpenSSL предоставляет средства для управления криптографическими ключами, что упрощает интеграцию защищенных соединений в общую архитектуру системы.

## 2 Конструкторский раздел

### 2.1 Функциональные требования к протоколу

Ниже перечислены функциональные требования к разрабатываемому протоколу.

1. Протокол должен поддерживать проведение одной конференции для произвольного количества участников.
2. Участники должны иметь возможность подключаться и отключаться от конференции во время ее проведения.

### 2.2 Роли участников конференции

При создании конференции участник, который ее создает (и отправляет приглашения остальным участникам) назначается управляющим конференцией.

Управляющий конференцией имеет возможность редактировать параметры конференции, рассылать приглашения, а также передать свои полномочия другому участнику конференции.

### 2.3 Типы сообщений

Разрабатываемый протокол предусматривает следующие типы передаваемых сообщений:

- **INVITE** — приглашение в конференцию. Получатель сообщения подключается к конференции, в которой участвует отправитель.
- **INVITE\_ACCEPT** — участник принимает приглашение.
- **INVITE\_REJECT** — участник отклоняет приглашение.
- **PART\_PRESENCE** — подключение/отключение участников конференции.
- **PART\_INFO** — изменение состояния отдельного участника конференции.
- **REENTER** — отправитель намеревается подключиться к существующей конференции (после потери соединения).
- **LEAVE** — отправитель намеривается покинуть конференцию. Получатели должны финализировать все ресурсы, связанные с этим участником и разорвать соединения. Без использования данного типа сообщения будет происходить ожидание переподключения участника.
- **AUDIO** — сообщение с аудио пакетом.
- **VIDEO** — сообщение с видео пакетом.

Схема типов сообщений представлена на рисунке 2.1.



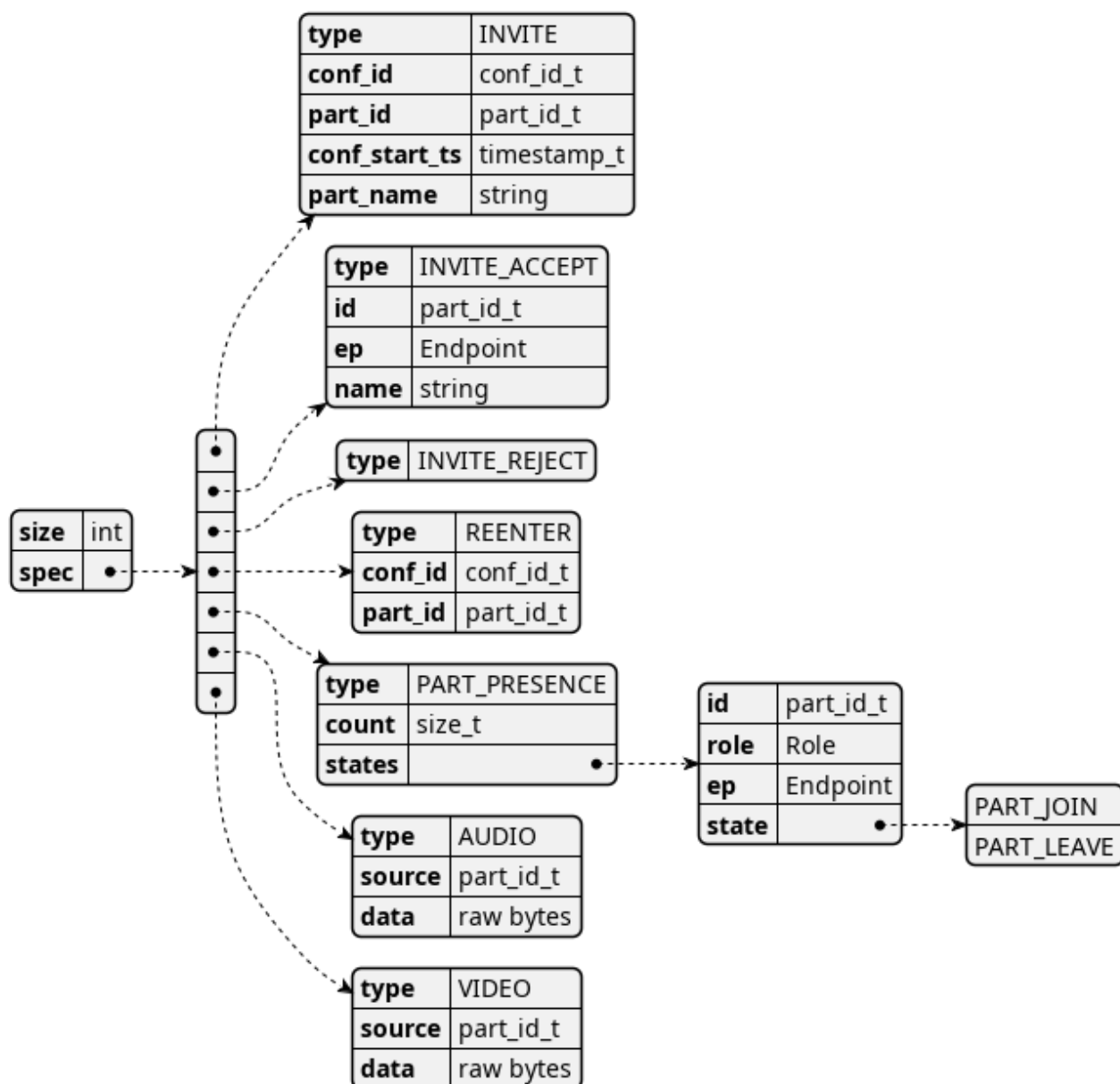


Рисунок 2.1 – Схема типов сообщений с полями

## 2.4 Медиа форматы

Снижение объемов передачи аудио/видео данных по сети требует использования кодеков, сжимающих медиа данные. Для разрабатываемого протокола применяются кодеки AAC [5] и H.264 [6]. Аудио кадр кодируется в стерео формате с частотой 48кГц. Видео кадр кодируется в разрешении 320 на 180 пикселей и частотой в 30 кадров в секунду.

## 2.5 Взаимодействия сторон

Рассмотрим сценарий с двумя участниками — «А» и «Б». Для организации конференции, участник «А» отправляет участнику «Б» сообщение типа INVITE. После чего участник «Б» должен ответить на него сообщением INVITE\_АССЕПТ для подключения к конферен-

ции участника «А». После успешного подключения, происходит инициализация каналов передачи медиа данных. Аудио и видео пакеты передаются с использованием сообщений типов AUDIO / VIDEO. Диаграмма последовательности описанного процесса представлена на рисунке 2.2.

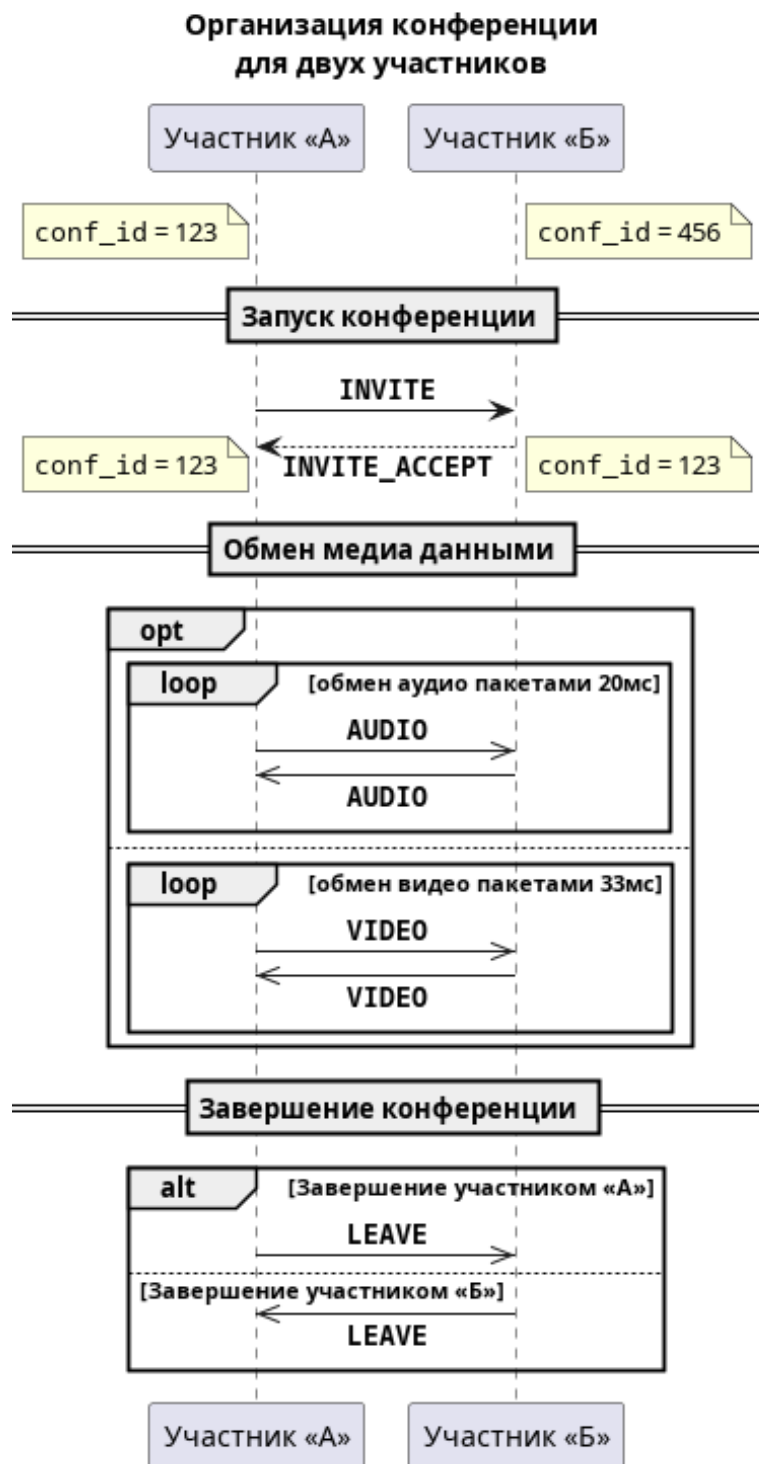


Рисунок 2.2 – Диаграмма последовательности для организации конференции для двух участников

Подключение последующих участников происходит аналогичным образом — один из

активных участников конференции отправляет сообщение с приглашением в конференцию новому участнику, и, после получения ответного сообщения **INVITE\_ACCEPT**, отправляет сообщение **PART\_PRESENCE** остальным участникам конференции для уведомления о подключении нового участника. Каждый из остальных участников, получив сообщение **PART\_PRESENCE** отправляет приглашение **INVITE** новому участнику для инициализации каналов связи между всеми парами участников конференции. Диаграмма последовательности описанного процесса представлена на рисунке 2.3.

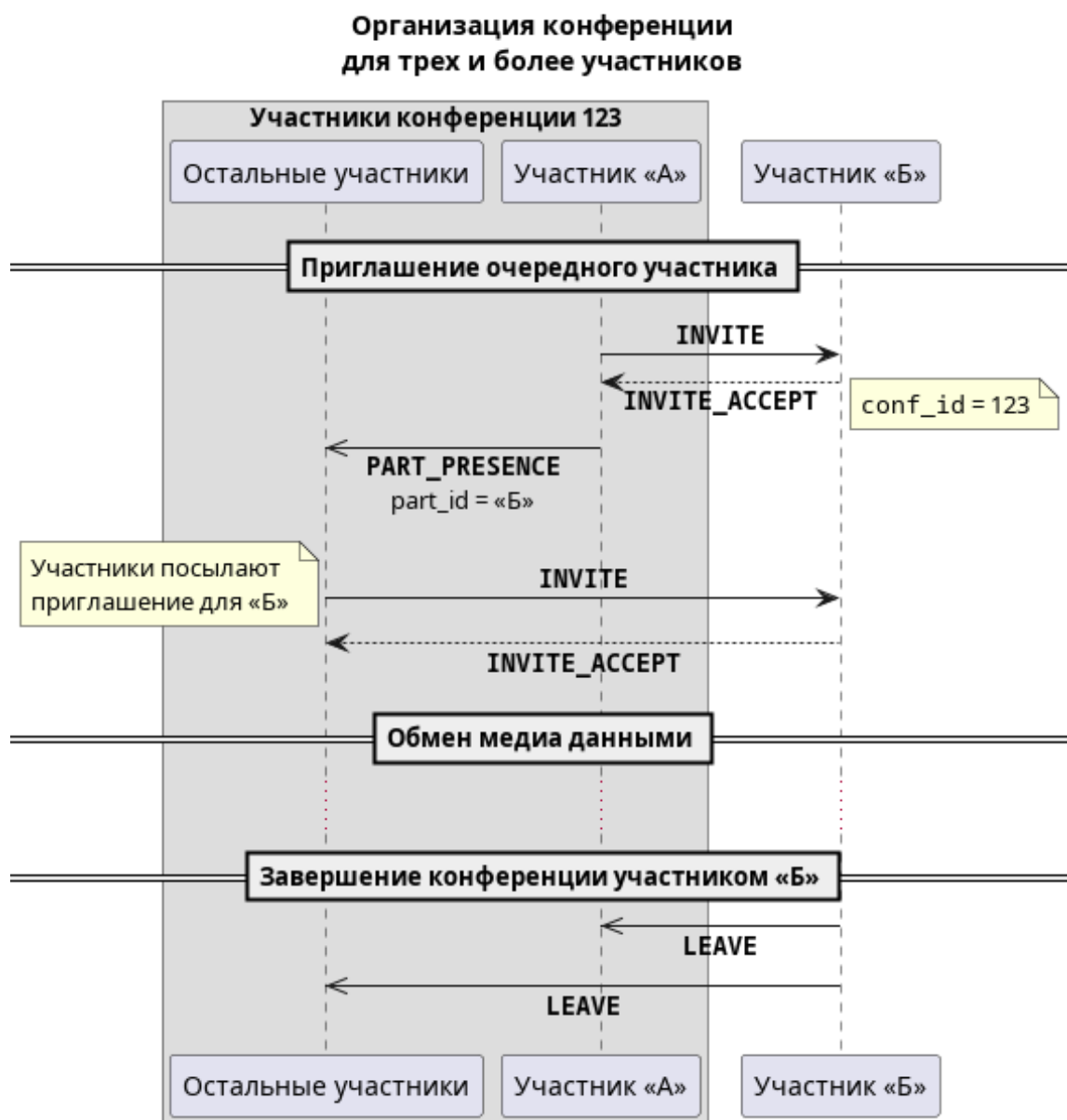


Рисунок 2.3 – Диаграмма последовательности для организации конференции для трех и более участников

## 2.6 Шифрование

Шифрование используется для обеспечения целостности и конфиденциальности передаваемых данных. (Защита от подслушивания, порчи и/или подделки сообщений). Проверка

сертификатов разрабатываемым протоколом не предусмотрена, но может быть проведена при повторном соединении с участником. В качестве протокола шифрования используется TLS v1.3 [7].

Выбор между протоколами TCP+TLS и SRTP для организации бессерверных видео конференций зависит от ряда факторов и требований. Рассмотрим преимущества и недостатки каждого из подходов:

Преимущества использования TCP+TLS:

- Надежность и безопасность — TCP обеспечивает надежную доставку данных, а TLS предоставляет шифрование и аутентификацию, защищая передаваемые данные.
- Совместимость — TCP и TLS являются широко распространенными и поддерживаемыми протоколами, что упрощает интеграцию с различными системами и платформами.
- Развитая инфраструктура — для TCP+TLS существует большое количество инструментов, библиотек и готовых решений, упрощающих разработку.

Недостатки:

- Более высокая задержка — TCP имеет больше накладных расходов по сравнению с UDP-протоколами, что может повлиять на качество аудио и видео.
- Более высокие требования к ресурсам — TCP-стек требует больше памяти и вычислительных ресурсов, что может быть критично для устройств с ограниченными возможностями.

Преимущества использования SRTP:

- Низкая задержка — SRTP, построенный поверх UDP, имеет меньше накладных расходов, что позволяет обеспечить более высокое качество аудио и видео.
- Меньшее потребление ресурсов — SRTP более эффективен в плане использования памяти и вычислительной мощности, что важно для мобильных устройств.
- Специализация на мультимедиа — SRTP разработан специально для защищенной передачи аудио и видео данных, в отличие от более универсального TCP+TLS.

Недостатки:

- Ограниченная совместимость — SRTP не так широко распространен, как TCP+TLS, что может усложнить интеграцию с существующими системами.
- Меньшая надежность — UDP, лежащий в основе SRTP, не гарантирует доставку данных, что может привести к потере пакетов в неблагоприятных сетевых условиях.
- Сложность реализации — SRTP требует реализации дополнительных механизмов для управления ключами шифрования, что усложняет разработку.

## 2.7 Переподключение

При потере соединения без получения сообщения **LEAVE** необходимо ожидать переподключение участника в течении 30 секунд. Если участник так и не подключился к конференции, то он считается вышедшим из конференции и в последующем сможет подключиться к ней только через приглашение от другого участника конференции.

Переподключение производится отправкой сообщения **REENTER** с идентификатором текущей конференции и идентификатором участника, потерявшего соединение. Если сообщение было получено по истечению 30 секундного интервала — оно отбрасывается, соединение закрывается и участник считается неподключившимся.

## 2.8 Определение качества каналов связи

Для реформирования топологии связей в бессерверных видео конференциях необходимо определить качество существующих каналов связи между участниками. Это может быть выполнено с использованием метрик, которые зависят от следующих параметров: временные метки отправки/приема аудио/видео пакета; размер пакета в байтах; количество переданных и повторно переданных пакетов.

На основе этих параметров могут быть вычислены такие показатели качества как:

- задержка передачи — временная разница между отправкой и получением пакета;
- джиттер — вариация задержки между последовательными пакетами;
- пропускная способность — объем данных, переданных за определенный период времени.

Используя данные метрики, можно оценить состояние каналов связи между участниками конференции и принять решение о необходимости реформирования топологии. Например, если канал между двумя участниками характеризуется высокой задержкой и потерями пакетов, можно рассмотреть вариант передачи медиа данных через третьего участника, который имеет более качественное соединение. Это позволит сократить нагрузку на перегруженный канал и переложить часть вычислительной нагрузки по перекодированию на более мощный клиент.

## 3 Технологический раздел

### 3.1 Средства разработки

Для реализации поддерживающего спроектированный протокол ПО был выбран язык С. В качестве средства сборки используется средство сборки CMake [8].

Для работы с медиа данными используется библиотека с открытым исходным кодом ffmpeg [9].

Шифрование сообщений осуществляется с использованием библиотеки OpenSSL.

Для автотестов используется фреймворк GoogleTest [10].

### 3.2 Листинги процедуры приглашения

Листинг 3.1 – Процедура приглашения со стороны приглашающего

```
// handshake procedure
//
// | step | client (inviter) | server (invitee) |
// | :--- | :----- | :----- |
// | 1    | sends invite msg | recvs invite msg |
// | 2    |                  | decides accept    |
// | 3    | recvs confirm msg | sends confirm msg |
static enum SError do_handshake_client(struct SConnection *con,
    struct SMessage *msg, struct SMsgInviteAccept **acceptMsg) {
    enum SError err = sconn_send(con, msg);
    if (err != SELECON_OK)
        return err;
    err = sconn_recv(con, &msg);
    if (err != SELECON_OK)
        message_free(&msg);
    else if (msg->type == SMSG_INVITE_ACCEPT)
        *acceptMsg = (struct SMsgInviteAccept *)msg;
    else {
        message_free(&msg);
        err = SELECON_INVITE_REJECTED;
    }
    return err;
}
```

Листинг 3.2 – Процедура приглашения со стороны приглашаемого

```
// handshake procedure
```

```

//
// | step | client (inviter) | server (invitee) |
// | :--- | :----- | :----- |
// | 1    | sends invite msg  | recvs invite msg  |
// | 2    |                   | decides accept    |
// | 3    | recvs confirm msg | sends confirm msg |
static enum SError do_handshake_srv(struct SParticipant *self,
    struct SEndpoint *listen_ep, struct SConnection *con,
    struct SMsgInvite **invite, invite_handler_fn_t handler) {
    struct SMessage *msg = NULL;
    enum SError err      = sconn_recv(con, &msg);
    if (err != SELECON_OK || msg->type != SMSG_INVITE)
        goto cleanup;
    bool accepted = handler((struct SMsgInvite *)msg);
    if (!accepted) {
        msg->size = sizeof(struct SMessage);
        msg->type = SMSG_INVITE_REJECT;
        err      = sconn_send(con, msg);
        goto cleanup;
    }
    *invite = (struct SMsgInvite *)msg;
    msg      = message_invite_accept_alloc(self->id, self->name,
        listen_ep);
    err      = sconn_send(con, msg);
cleanup:
    message_free(&msg);
    return err;
}

```

Листинг 3.3 – Воркер, обрабатывающий входящие приглашения в конференции

```

static void *invite_worker(void *arg) {
    struct SContext *ctx          = arg;
    struct SConnection *listen_con = NULL;
    enum SError err              = sconn_listen(&listen_con,
        &ctx->listen_ep);
    while (ctx->initialized && (err == SELECON_OK || err ==
        SELECON_CON_TIMEOUT)) {
        struct SConnection *con = NULL;

```

```

err                                = sconn_accept_secure(listen_con,
    &con, SELECON_DEFAULT_SECURE_TIMEOUT);
if (err == SELECON_OK) {
    struct SMsgInvite *invite = NULL;
    err = do_handshake_srv(&ctx->self, &ctx->listen_ep, con,
        &invite, ctx->invite_handler);
    if (err == SELECON_OK && invite != NULL)
        err = handle_invite(ctx, con, invite);
    if (err != SELECON_OK || invite == NULL)
        sconn_disconnect(&con);
    message_free((struct SMessage *)&invite);
}
}
sconn_disconnect(&listen_con);
if (ctx->initialized && err != SELECON_OK)
    printf("listener error: %s\n", serror_str(err));
return NULL;
}

```

Листинг 3.4 – Процедура приглашения участника в конференцию

```

static enum SError invite_connected(struct SContext *context,
    struct SConnection *con, part_id_t *out_part_id) {
    struct SMessage *inviteMsg = message_invite_alloc(
        context->conf_id, context->conf_start_ts,
        context->self.id, context->self.name);
    struct SMsgInviteAccept *acceptMsg = NULL;
    enum SError err = do_handshake_client(con,
        inviteMsg, &acceptMsg);
    if (err != SELECON_OK || acceptMsg == NULL)
        return err;
    // send other participants info about invitee
    struct SMsgPartPresence *msg = message_part_presence_alloc(1);
    msg->states[0].id = acceptMsg->id;
    msg->states[0].ep = acceptMsg->ep;
    msg->states[0].role = SROLE_CLIENT;
    msg->states[0].state = PART_JOIN;
    pthread_rwlock_wrlock(&context->part_rwlock);
    for (size_t i = 0; i < context->nb_participants - 1; ++i)

```



```

    sconn_send(context->participants[i].connection, (struct
        SMessage *)msg);
add_participant(context, acceptMsg->id, acceptMsg->name, con);
if (context->nb_participants == 2 &&
    !context->conf_thread_working) {
    pthread_create(&context->conf_thread, NULL, conf_worker,
        context);
    pthread_setname_np(context->conf_thread, "conf");
}
pthread_rwlock_unlock(&context->part_rwlock);
*out_part_id = acceptMsg->id;
message_free((struct SMessage **)&msg);
message_free((struct SMessage **)&acceptMsg);
return SELECON_OK;
}

```

### 3.3 Сборка и запуск

Сборка основного проекта осуществляется с использованием утилиты `cmake`:

```

mkdir build
cd build
cmake ..
make цель

```

где «цель» может быть `selecon_cli` для сборки консольной утилиты, либо `unittests` для сборки тестов.

Консольная утилита имеет следующие параметры запуска:

```

$ ./selecon_cli --help
./selecon_cli [OPTIONS]

```

example demo application for selecon protocol

OPTIONS:

```

-h|--help          show this message
-l|--listen-on address current participant address (default 0.0.0.0:11235)
-u|--user username set user name
--version          print version and exit
--stub filename    stream given media file in a loop

```

#### DESCRIPTION:

Address can be IPv4/IPv6 (eg: 192.168.100.1:11235) or socket file path in format file://<os-path-to-socket>.

Список команд, поддерживаемых утилитой:

```
$ ./selecon_cli
> help
list of available commands:
dev      manage IO devices
dump     print info about current selecon context state
exit     end active conference and close cli tool
help     show this message
invite   send invitation for joining active conference to other client
leave    exit conference without exiting cli tool
quit     same as exit
say      send text message to conference chat
sleep    sleep
stub     set stub media file for playing in conference
```

В качестве видео заглушки можно использовать видео файлы в формате MP4.

## 3.4 Тестирование ПО

Ниже приведен список автотестов, разработанных для проверки реализованного протокола.

- Подключение второго участника к конференции, второй участник отклонил приглашение.
- Подключение второго участника к конференции, второй участник принял приглашение.
- Отправка аудио данных от одного участника другому.
- Отправка видео данных от одного участника другому.
- Подключение четырех участников к конференции.
- Отправка аудио данных от одного участника к другим трем.
- Отправка видео данных от одного участника к другим трем.

## 3.5 Трасировка пакетов

На рисунке 3.1 представлен сниф пакетов с использованием программы wireshark [11] во время инициализации конференции с двумя участниками. На рисунке 3.2 представлен сниф того же сценария, но с использованием TLS протокола.

No.	Time	Source	Destination	Protocol	Length	Info
71	75.5749776...	127.0.0.1	127.0.0.1	TCP	74	43112 → 4647 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1415507521 TSecr=0 WS=
72	75.5750114...	127.0.0.1	127.0.0.1	TCP	74	4647 → 43112 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=1415507521
73	75.5750338...	127.0.0.1	127.0.0.1	TCP	66	43112 → 4647 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1415507521 TSecr=1415507521
74	75.5750964...	127.0.0.1	127.0.0.1	TCP	100	43112 → 4647 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=34 TSval=1415507521 TSecr=1415507521
75	75.5751116...	127.0.0.1	127.0.0.1	TCP	66	4647 → 43112 [ACK] Seq=1 Ack=35 Win=65536 Len=0 TSval=1415507521 TSecr=1415507521
76	75.5751704...	127.0.0.1	127.0.0.1	TCP	208	4647 → 43112 [PSH, ACK] Seq=1 Ack=35 Win=65536 Len=142 TSval=1415507521 TSecr=1415507521
77	75.5751890...	127.0.0.1	127.0.0.1	TCP	66	43112 → 4647 [ACK] Seq=35 Ack=143 Win=65408 Len=0 TSval=1415507521 TSecr=1415507521

Рисунок 3.1 – Сниф wireshark соединения двух участников без использования TLS

No.	Time	Source	Destination	Protocol	Length	Info
9	5.052357137	127.0.0.1	127.0.0.1	TCP	74	47572 → 4647 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1407281997 TSecr=0
10	5.052389140	127.0.0.1	127.0.0.1	TCP	74	4647 → 47572 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=1407281997
11	5.052408717	127.0.0.1	127.0.0.1	TCP	66	47572 → 4647 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1407281997 TSecr=1407281997
12	5.057595809	127.0.0.1	127.0.0.1	TLSv1.3	363	Client Hello
13	5.057603044	127.0.0.1	127.0.0.1	TCP	66	4647 → 47572 [ACK] Seq=1 Ack=298 Win=65280 Len=0 TSval=1407282002 TSecr=1407282002
14	5.065332087	127.0.0.1	127.0.0.1	TLSv1.3	2401	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data,
15	5.065341389	127.0.0.1	127.0.0.1	TCP	66	47572 → 4647 [ACK] Seq=298 Ack=2336 Win=63744 Len=0 TSval=1407282010 TSecr=1407282010
16	5.066235572	127.0.0.1	127.0.0.1	TLSv1.3	146	Change Cipher Spec, Application Data
17	5.066372141	127.0.0.1	127.0.0.1	TLSv1.3	321	Application Data
18	5.066378379	127.0.0.1	127.0.0.1	TLSv1.3	130	Application Data
19	5.066421656	127.0.0.1	127.0.0.1	TLSv1.3	321	Application Data
20	5.107172277	127.0.0.1	127.0.0.1	TCP	66	47572 → 4647 [ACK] Seq=442 Ack=2846 Win=65536 Len=0 TSval=1407282052 TSecr=1407282011
21	5.107191376	127.0.0.1	127.0.0.1	TLSv1.3	230	Application Data
22	5.107201599	127.0.0.1	127.0.0.1	TCP	66	47572 → 4647 [ACK] Seq=442 Ack=3010 Win=65408 Len=0 TSval=1407282052 TSecr=1407282052

Рисунок 3.2 – Сниф wireshark соединения двух участников с использованием TLS

На рисунках 3.3–3.4 представлены снифы пакетов во время инициализации конференции с тремя участниками. По рисунку видны 3 этапа соединения: инициализация конференции двумя участниками; отправка и получение приглашения для третьего участника и информирование второго участника о новом подключившемся участнике; рукопожатие второго и третьего участников.

No.	Time	Source	Destination	Protocol	Length	Info
15	11.1488...	127.0.0.1	127.0.0.2	TCP	74	44060 → 4647 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=305237562 TSecr=0 WS=128
16	11.1488...	127.0.0.2	127.0.0.1	TCP	74	4647 → 44060 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2530500019 TSe
17	11.1489...	127.0.0.1	127.0.0.2	TCP	66	44060 → 4647 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=305237562 TSecr=2530500019
18	11.1490...	127.0.0.1	127.0.0.2	TCP	108	44060 → 4647 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=42 TSval=305237562 TSecr=2530500019
19	11.1490...	127.0.0.2	127.0.0.1	TCP	66	4647 → 44060 [ACK] Seq=1 Ack=43 Win=65536 Len=0 TSval=2530500019 TSecr=305237562
20	11.1490...	127.0.0.2	127.0.0.1	TCP	208	4647 → 44060 [PSH, ACK] Seq=1 Ack=43 Win=65536 Len=142 TSval=2530500019 TSecr=305237562
21	11.1490...	127.0.0.1	127.0.0.2	TCP	66	44060 → 4647 [ACK] Seq=43 Ack=143 Win=65408 Len=0 TSval=305237562 TSecr=2530500019
26	19.5150...	127.0.0.1	127.0.0.3	TCP	74	57888 → 4903 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=4111713139 TSecr=0 WS=128
27	19.5150...	127.0.0.3	127.0.0.1	TCP	74	4903 → 57888 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2000629589 TSe
28	19.5150...	127.0.0.1	127.0.0.3	TCP	66	57888 → 4903 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4111713139 TSecr=2000629589
29	19.5150...	127.0.0.1	127.0.0.3	TCP	108	57888 → 4903 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=42 TSval=4111713139 TSecr=2000629589
30	19.5151...	127.0.0.3	127.0.0.1	TCP	66	4903 → 57888 [ACK] Seq=1 Ack=43 Win=65536 Len=0 TSval=2000629589 TSecr=4111713139
31	19.5151...	127.0.0.3	127.0.0.1	TCP	208	4903 → 57888 [PSH, ACK] Seq=1 Ack=43 Win=65536 Len=142 TSval=2000629589 TSecr=4111713139
32	19.5151...	127.0.0.1	127.0.0.3	TCP	66	57888 → 4903 [ACK] Seq=43 Ack=143 Win=65408 Len=0 TSval=4111713139 TSecr=2000629589
33	19.5152...	127.0.0.1	127.0.0.2	TCP	218	44060 → 4647 [PSH, ACK] Seq=43 Ack=143 Win=65536 Len=152 TSval=305245928 TSecr=2530500019
34	19.5181...	127.0.0.1	127.0.0.3	TCP	74	57896 → 4903 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=4111713142 TSecr=0 WS=128
35	19.5181...	127.0.0.3	127.0.0.1	TCP	74	4903 → 57896 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2000629592 TSe
36	19.5181...	127.0.0.1	127.0.0.3	TCP	66	57896 → 4903 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4111713142 TSecr=2000629592
37	19.5181...	127.0.0.1	127.0.0.3	TCP	108	57896 → 4903 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=42 TSval=4111713142 TSecr=2000629592
38	19.5182...	127.0.0.3	127.0.0.1	TCP	66	4903 → 57896 [ACK] Seq=1 Ack=43 Win=65536 Len=0 TSval=2000629592 TSecr=4111713142
39	19.5183...	127.0.0.3	127.0.0.1	TCP	208	4903 → 57896 [PSH, ACK] Seq=1 Ack=43 Win=65536 Len=142 TSval=2000629592 TSecr=4111713142
40	19.5183...	127.0.0.1	127.0.0.3	TCP	66	57896 → 4903 [ACK] Seq=43 Ack=143 Win=65408 Len=0 TSval=4111713142 TSecr=2000629592
41	19.5184...	127.0.0.2	127.0.0.1	TCP	218	4647 → 44060 [PSH, ACK] Seq=143 Ack=195 Win=65536 Len=152 TSval=2530508388 TSecr=305245928
42	19.5184...	127.0.0.1	127.0.0.2	TCP	66	44060 → 4647 [ACK] Seq=195 Ack=295 Win=65408 Len=0 TSval=305245931 TSecr=2530508388

Рисунок 3.3 – Сниф wireshark соединения трех участников без использования TLS

No.	Time	Source	Destination	Protocol	Leng	Info
49	51.3493...	127.0.0.1	127.0.0.2	TCP	74	41754 → 5159 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=306228764 TSecr=0 WS=
50	51.3493...	127.0.0.2	127.0.0.1	TCP	74	5159 → 41754 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2531491221
51	51.3493...	127.0.0.1	127.0.0.2	TCP	66	41754 → 5159 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=306228764 TSecr=2531491221
52	51.3545...	127.0.0.1	127.0.0.2	TLSv1.3	363	Client Hello
53	51.3545...	127.0.0.2	127.0.0.1	TCP	66	5159 → 41754 [ACK] Seq=1 Ack=298 Win=65280 Len=0 TSval=2531491226 TSecr=306228769
54	51.3663...	127.0.0.2	127.0.0.1	TLSv1.3	24...	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data,
55	51.3663...	127.0.0.1	127.0.0.2	TCP	66	41754 → 5159 [ACK] Seq=298 Ack=2336 Win=63744 Len=0 TSval=306228781 TSecr=2531491238
56	51.3675...	127.0.0.1	127.0.0.2	TLSv1.3	146	Change Cipher Spec, Application Data
57	51.3677...	127.0.0.2	127.0.0.1	TLSv1.3	321	Application Data
58	51.3677...	127.0.0.1	127.0.0.2	TLSv1.3	130	Application Data
59	51.3678...	127.0.0.2	127.0.0.1	TLSv1.3	321	Application Data
60	51.4104...	127.0.0.1	127.0.0.2	TCP	66	41754 → 5159 [ACK] Seq=442 Ack=2846 Win=65536 Len=0 TSval=306228825 TSecr=2531491239
61	51.4104...	127.0.0.2	127.0.0.1	TLSv1.3	230	Application Data
62	51.4105...	127.0.0.1	127.0.0.2	TCP	66	41754 → 5159 [ACK] Seq=442 Ack=3010 Win=65408 Len=0 TSval=306228825 TSecr=2531491282
67	58.6641...	127.0.0.1	127.0.0.3	TCP	74	33818 → 5671 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=4112703289 TSecr=0 WS=
68	58.6641...	127.0.0.3	127.0.0.1	TCP	74	5671 → 33818 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2001619740
69	58.6641...	127.0.0.1	127.0.0.3	TCP	66	33818 → 5671 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4112703290 TSecr=2001619740
70	58.6651...	127.0.0.1	127.0.0.3	TLSv1.3	363	Client Hello
71	58.6651...	127.0.0.3	127.0.0.1	TCP	66	5671 → 33818 [ACK] Seq=1 Ack=298 Win=65280 Len=0 TSval=2001619740 TSecr=4112703290
72	58.6802...	127.0.0.3	127.0.0.1	TLSv1.3	24...	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data,
73	58.6802...	127.0.0.1	127.0.0.3	TCP	66	33818 → 5671 [ACK] Seq=298 Ack=2336 Win=63744 Len=0 TSval=4112703306 TSecr=2001619756
74	58.6809...	127.0.0.1	127.0.0.3	TLSv1.3	146	Change Cipher Spec, Application Data
75	58.6810...	127.0.0.3	127.0.0.1	TLSv1.3	321	Application Data
76	58.6810...	127.0.0.1	127.0.0.3	TLSv1.3	130	Application Data
77	58.6810...	127.0.0.3	127.0.0.1	TLSv1.3	321	Application Data
78	58.7222...	127.0.0.1	127.0.0.3	TCP	66	33818 → 5671 [ACK] Seq=442 Ack=2846 Win=65536 Len=0 TSval=4112703348 TSecr=2001619756
79	58.7222...	127.0.0.3	127.0.0.1	TLSv1.3	230	Application Data
80	58.7222...	127.0.0.1	127.0.0.3	TCP	66	33818 → 5671 [ACK] Seq=442 Ack=3010 Win=65408 Len=0 TSval=4112703348 TSecr=2001619798
81	58.7223...	127.0.0.1	127.0.0.2	TLSv1.3	240	Application Data
82	58.7247...	127.0.0.1	127.0.0.3	TCP	74	33822 → 5671 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=4112703350 TSecr=0 WS=
83	58.7247...	127.0.0.3	127.0.0.1	TCP	74	5671 → 33822 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2001619800
84	58.7248...	127.0.0.1	127.0.0.3	TCP	66	33822 → 5671 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4112703350 TSecr=2001619800
85	58.7260...	127.0.0.1	127.0.0.3	TLSv1.3	363	Client Hello
86	58.7260...	127.0.0.3	127.0.0.1	TCP	66	5671 → 33822 [ACK] Seq=1 Ack=298 Win=65280 Len=0 TSval=2001619801 TSecr=4112703351
87	58.7409...	127.0.0.3	127.0.0.1	TLSv1.3	24...	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data,
88	58.7410...	127.0.0.1	127.0.0.3	TCP	66	33822 → 5671 [ACK] Seq=298 Ack=2336 Win=63744 Len=0 TSval=4112703366 TSecr=2001619816
89	58.7422...	127.0.0.1	127.0.0.3	TLSv1.3	146	Change Cipher Spec, Application Data
90	58.7425...	127.0.0.3	127.0.0.1	TLSv1.3	321	Application Data
91	58.7425...	127.0.0.1	127.0.0.3	TLSv1.3	130	Application Data
92	58.7425...	127.0.0.3	127.0.0.1	TLSv1.3	321	Application Data
93	58.7662...	127.0.0.2	127.0.0.1	TCP	66	5159 → 41754 [ACK] Seq=3010 Ack=616 Win=65536 Len=0 TSval=2531498638 TSecr=306236137
94	58.7862...	127.0.0.1	127.0.0.3	TCP	66	33822 → 5671 [ACK] Seq=442 Ack=2846 Win=65536 Len=0 TSval=4112703412 TSecr=2001619818
95	58.7862...	127.0.0.3	127.0.0.1	TLSv1.3	230	Application Data
96	58.7862...	127.0.0.1	127.0.0.3	TCP	66	33822 → 5671 [ACK] Seq=442 Ack=3010 Win=65408 Len=0 TSval=4112703412 TSecr=2001619862
97	58.7864...	127.0.0.2	127.0.0.1	TLSv1.3	240	Application Data
98	58.7864...	127.0.0.1	127.0.0.2	TCP	66	41754 → 5159 [ACK] Seq=616 Ack=3184 Win=65408 Len=0 TSval=306236201 TSecr=2531498658

Рисунок 3.4 – Снимок wireshark соединения трех участников с использованием TLS

## 4 Исследовательский раздел

### 4.1 Постановка исследования

Цель исследования — выявление зависимости задержки аудио и видео, а также общей нагрузки на процессор от числа участников в конференции.

Исследование проведено на ноутбуке Intel i5-8365U 4.1GHz, 16GB оперативной памяти, под операционной системой Debian 12.

Количество участников варьируется от 2 до 7, время конференции — 1 минута. Во время конференции все участники транслировали одно и то же видео.

### 4.2 Результаты

Результаты представлены на графике 4.1.

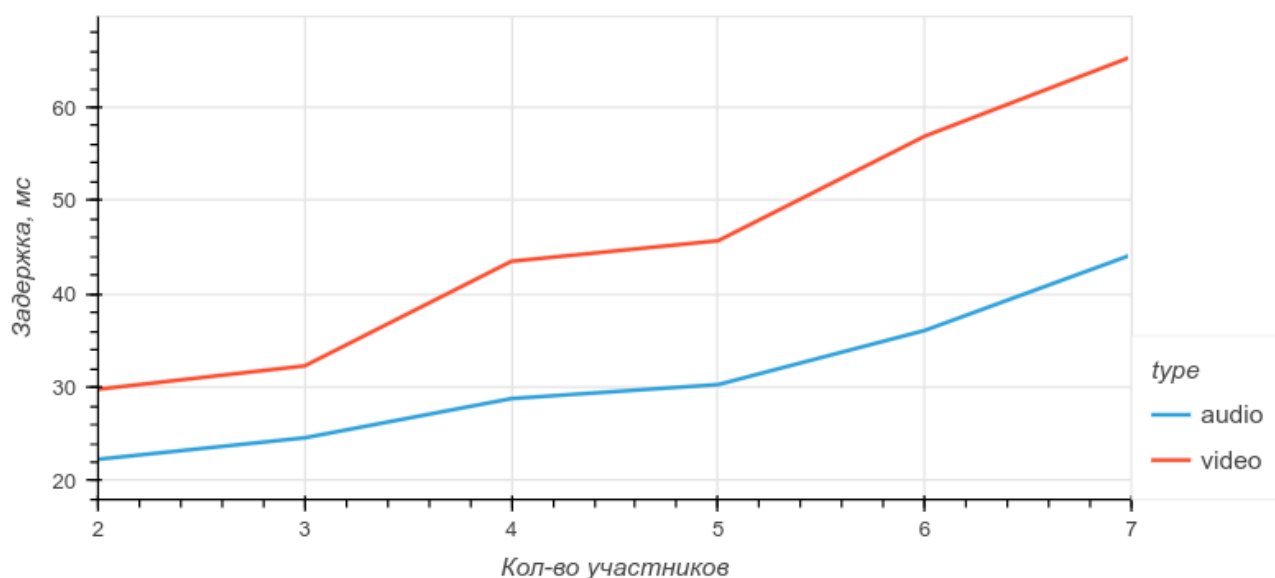


Рисунок 4.1 – Зависимость средней задержки аудио и видео от числа участников конференции

На основе полученных результатов можно предположить, что увеличение задержки связано с увеличивающейся нагрузкой на процессор при увеличении количества участников.

## ЗАКЛЮЧЕНИЕ

В результате работы был спроектирован протокол бессерверных конференций. Разработана программная реализация спроектированного протокола и реализовано ПО, осуществляющее пересылку видео и аудио потоков в рамках бессерверной конференции. Проведено тестирование и исследование ПО. В результате исследования была выявлена зависимость задержки видео и аудио потоков от количества участников конференции.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. RTP: A Transport Protocol for Real-Time Applications / H. Schulzrinne [и др.]. — 07.2003. — DOI: 10.17487/RFC3550. — URL: <https://www.rfc-editor.org/info/rfc3550>. RFC 3550.
2. SIP: Session Initiation Protocol / E. Schooler [и др.]. — 07.2002. — DOI: 10.17487/RFC3261. — URL: <https://www.rfc-editor.org/info/rfc3261>. RFC 3261.
3. *Agboh C., Schulzrinne H.* Session Initiation Protocol (SIP)-H.323 Interworking Requirements. — 07.2005. — DOI: 10.17487/RFC4123. — URL: <https://www.rfc-editor.org/info/rfc4123>. RFC 4123.
4. OpenSSL — <https://www.openssl.org/> (дата обращения: 10.12.2024). — [Электронный ресурс].
5. RTP Payload Format for MPEG-4 Audio/Visual Streams / Y. Matsui [и др.]. — 11.2000. — DOI: 10.17487/RFC3016. — URL: <https://www.rfc-editor.org/info/rfc3016>. RFC 3016.
6. H.264: Advanced video coding for generic audiovisual services — <https://www.itu.int/rec/T-REC-H.264> (дата обращения: 10.12.2024). — (Электронный ресурс).
7. *Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.3. — 08.2018. — DOI: 10.17487/RFC8446. — URL: <https://www.rfc-editor.org/info/rfc8446>. RFC 8446.
8. CMake — Upgrade Your Software Build System — <https://cmake.org/> (дата обращения: 20.12.2024). — (Электронный ресурс).
9. Ffmpeg — <https://www.ffmpeg.org/> (дата обращения: 21.12.2024). — (Электронный ресурс).
10. GoogleTest User's Guide — <https://google.github.io/googletest/> (дата обращения: 21.12.2024). — (Электронный ресурс).
11. Wireshark · Go Deep — <https://www.wireshark.org/> (дата обращения: 23.12.2024). — (Электронный ресурс).