МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА СИСТЕМ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ

**ТЕХНОЛОГИИ ПОТОКОВОЙ ПЕРЕДАЧИ ИНФОРМАЦИИ В СЕТИ ИНТЕРНЕТ. ПРОТОКОЛ WEBRTC И ЕГО УСТРОЙСТВО.**

Реферат по дисциплине

«Веб-технологии»

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИТБ-5301-02-20 | \_\_\_\_\_\_\_\_\_\_\_  (подпись) | / Червоткин Д.А. / |
| Проверил: | \_\_\_\_\_\_\_\_\_\_\_  (подпись) | / Земцов М.А. / |
|  |  |  |

Киров 2023

**Содержание**

[Введение 3](#_Toc153803002)

[1 Общие принципы потоковой передачи 4](#_Toc153803003)

[2 Протокол WebRTC 6](#_Toc153803004)

[2.1 Общий принцип действия WebRTC 6](#_Toc153803005)

[2.2 Сигнализация 6](#_Toc153803006)

[2.3 Установка соединения 7](#_Toc153803007)

[2.4 Обеспечение безопасности 7](#_Toc153803008)

[2.5 Коммуникация 8](#_Toc153803009)

[Заключение 10](#_Toc153803010)

[Приложение А 11](#_Toc153803011)

# **Введение**

Потоковая передача данных - процесс постоянного сбора информации в момент её создания и передвижения. Эта информация обрабатывается программным обеспечением для потоковой обработки. Потоковая передача данных в сочетании с обработкой позволяет получать интеллектуальные данные в реальном времени.

Потоки данных могут быть созданы из различных источников в любом формате и объеме. Самые мощные потоки объединяют несколько источников, чтобы получить полную картину действий и процессов.

К примеру, данные из сети, серверов и приложений могут быть объединены для мониторинга работоспособности вашего веб-сайта и обнаружения ухудшения производительности.

Один из яркий примеров потоковой передачи данных – протокол WebRTC.

WebRTC (Web Real-Time Communication — коммуникация в режиме реального времени) — это API (Application Programming Interface — программный интерфейс приложения) и протокол. Протокол WebRTC — это набор правил, позволяющий двум агентам WebRTC (браузерам) вести двунаправленную (bi-directional) безопасную коммуникацию в реальном времени. Среди особенностей данного протокола можно выделить следующие:

* Открытый стандарт;
* Разные реализации;
* Доступность в браузерах;
* Обязательное шифрование;
* Отображение NAT (NAT Traversal);
* Перепрофилирование существующих технологий;
* Контроль перегрузки (congestion control);
* Низкая задержка (на уровне долей секунды, sub-second latency).

# **1 Общие принципы потоковой передачи**

Потоковые данные – это данные, которые непрерывным и инкрементальным образом передаются в больших объемах, при этом на обработку затрачивается как можно меньше времени. Организации располагают тысячами источников данных, одновременно передающих сообщения, записи или данные, размер которых может варьироваться в пределах от нескольких байт до нескольких мегабайт (МБ).

Поток данных обладает следующими особенностями:

* Хронологическая значимость;
* Непрерывность потока;
* Уникальность;
* Отсутствие гомогенности;
* Неидеальность.

Хронологическая значимость подразумевает, что индивидуальные элементы потока данных содержат временные метки. Сам по себе поток данных может быть чувствительным к временным параметрам и обладать уменьшающейся значимостью по истечении определенного интервала времени. Например, если говорить о передаче видео, важно, чтобы получатель (зритель) получал видеокадры с как можно меньшей задержкой относительно ведущего.

Непрерывность потока подразумевает, что потоки данных не обладают началом или концом. Они собирают данные на постоянной основе и существуют непрерывно на протяжении всего необходимого времени.

Повторная передача потока данных вызывает проблемы из-за чувствительности к временным параметрам. Таким образом, задача точной обработки данных в режиме реального времени обретает критическую важность. К сожалению, возможности по предоставлению повторных передач ограничены в рамках большинства источников потоковой передачи данных.

Отсутствие гомогенности подразумевает, что некоторые источники могут выполнять потоковую передачу данных во множестве структурированных форматов, таких как JSON, Avro и списки разделенных запятыми значений – CSV, используя типы данных, которые состоят из строк, чисел, дат и двоичных параметров. Системы обработки потоков должны обладать возможностями по обслуживанию различных вариантов данных.

Неидеальность подразумевает, что временные ошибки в источнике могут приводить к повреждениям или утере элементов потоковых данных. Задачи по гарантированию согласованности данных могут оказаться довольно сложными из-за непрерывной природы потока. Системы аналитики и обработки потоков обычно обладают логикой для подтверждения данных, что позволяет минимизировать количество ошибок либо смягчать их влияние.

Система обработки потока является предпочтительной для большинства сценариев использования, подразумевающих непрерывное формирование новых и динамических данных. Обработка потоковых данных применима в большинстве отраслевых сегментов и случаев использования, подразумевающих обработку больших данных.

В качестве примера можно привести:

* Анализ данных;
* Приложения Интернета вещей;
* Финансовая аналитика;
* Рекомендации насчёт режима реального времени;
* Гарантии обслуживания;
* Мультимедиа и игры.

# **2 Протокол WebRTC**

## **2.1 Общий принцип действия WebRTC**

Процесс установки соединения делится на 4 этапа:

* Сигнализация (signalling);
* Подключение (установка соединения) (connection);
* Безопасность (securing);
* Коммуникация (взаимодействие) (communication).

При этом для перехода на следующий этап обязательно успешное завершение предыдущего. Каждый этап – комбинация других протоколов. Иными словами, WebRTC – комбинация хорошо известных технологий, появившихся на момент появления протокола в начале 2000-х годов.

## **2.2 Сигнализация**

При запуске WebRTC агент не знает, с кем и по поводу чего будет происходит коммуникация. И именно сигнализация решает данную проблему. Это первый этап и его назначение – подготовка вызова (звонка) (call) для того, чтобы два агента WebRTC могли начать коммуникацию.

Сигнализация осуществляется с помощью существующего протокола SDP (Session Description Protocol — протокол описания сессии). SDP – это текстовый протокол. Каждое сообщение SDP состоит из нескольких пар ключ/значение и содержит список "медиа разделов" (media sections). SDP, которыми обмениваются агенты WebRTC, содержит следующую информацию:

* IP-адреса и порты, по которым можно получить доступ к агенту (candidates — кандидаты)
* какое количество аудио и видео треков хочет отправить агент;
* какие аудио и видео кодеки поддерживаются каждым агентом;
* значения, используемые в процессе подключения (uFrag/uPwd);
* значения, используемые для обеспечения безопасности (certificate fingerprint — отпечаток сертификата).

WebRTC, обычно, не используется для передачи сигнальных сообщений (signalling messages). Для передачи SDP между подключенными пирами могут использоваться такие технологии, как конечные точки REST, веб-сокеты или прокси аутентификации (authentication proxies).

## **2.3 Установка соединения**

В процессе сигнализации агенты WebRTC получают достаточно информации для того, чтобы попытаться выполнить подключение. Для этого используется другая технология под названием ICE.

ICE (Interactive Connectivity Establishment — установка интерактивного соединения) — это еще один протокол, предшествующий появлению WebRTC. ICE позволяет устанавливать соединение между двумя агентами. Агенты могут находиться в одной (локальной) сети или в разных концах света. ICE — это решение для установки прямого соединения без центрального сервера.

После успешного подключения ICE WebRTC приступает к установке зашифрованного транспортного канала. Он используется для передачи аудио, видео и других данных.

## **2.4 Обеспечение безопасности**

После создания двунаправленного канала коммуникации (с помощью ICE), необходимо сделать этот канал безопасным. Это делается с помощью двух протоколов, также разработанных задолго до появления WebRTC. Первый протокол — это DTLS (Datagram Transport Layer Security — протокол датаграмм безопасности транспортного уровня), который является просто TLS поверх UDP. TLS — это криптографический протокол, который используется для безопасной коммуникации через HTTPS. Второй протокол — это SRTP (Secure Real-time Transport Protocol — используется для безопасной передачи данных в реальном времени).

Сначала WebRTC выполняет рукопожатие (handshake) DTLS с помощью соединения ICE. В отличие от HTTPS, WebRTC не использует центральный орган (central authority) для проверки сертификатов. Вместо этого WebRTC проверяет, что сертификат, переданный через DTLS, совпадает с отпечатком, переданным в процессе сигнализации. В дальнейшем DTLS-подключение используется для передачи сообщений по DataChannel (каналу передачи данных).

Для передачи аудио/видео используется другой протокол под названием RTP (Real-time Transport Protocol — протокол передачи данных в реальном времени). Пакеты, передаваемые по RTP, защищаются с помощью SRTP. Сессия SRTP начинается с извлечения ключей из установленной сессии DTLS.

Если предыдущие этапы завершились успешно, имеется двунаправленная и безопасная коммуникация. Если соединение между агентами WebRTC является стабильным, можно приступать к обмену данными. Однако в реальном мире постоянно приходится сталкиваться с потерей пакетов и ограниченной пропускной способностью.

## **2.5 Коммуникация**

Итак, безопасное двунаправленное соединение между двумя агентами WebRTC установлено, возможен обмен данными. Для этого используется два протокола: RTP и SCTP (Stream Control Transmission Protocol — протокол передачи с управлением потоком). RTP используется для передачи медиа, зашифрованного с помощью SRTP, а SCTP — нужен для отправки и приема сообщений по DataChannel, зашифрованных с помощью DTLS.

RTP является довольно минималистичным, но он предоставляет все необходимое для потоковой передачи данных в реальном времени. Гибкость RTP позволяет разработчикам решать проблемы, связанные с задержкой, потерей данных и перегрузкой, самыми разнообразными способами.

Последним протоколом в стеке является SCTP. Он предоставляет множество настроек, связанных с доставкой сообщений. Например, возможно пожертвовать надежностью и правильным порядком доставки пакетов данных в пользу низкой задержки доставки. Именно она является критически важной для коммуникации в реальном времени.

# **Заключение**

Потоковая передача данных – очень важное направление в информационных технологиях. Однако архитектура потоковой передачи требует особой подготовки, обусловленной характером и объемом данных.

Приложения потоковой трансляции данных требует стабильности работы, низкой задержки и высокой доступности. Потребители регулярно получают новые данные из потоков для последующей обработки. Задержки на стороне поставщика данных могут вызвать различные ошибки.

Одно из решений для потоковой передачи – протокол WebRTC, объединяющий в себе множество ранее существовавших протоколов. Таким образом можно предположить, что данный протокол является оркестратором большого количества других протоколов.

# **Приложение А**

(справочное)

**Библиографический список**

1. Что такое потоковая передача данных? [Электронный ресурс]. URL: <https://aws.amazon.com/ru/what-is/streaming-data/> (дата обращения – 18.12.2023)
2. WebRTC для всех и каждого. Часть 1 [Электронный ресурс]. URL: <https://habr.com/ru/companies/timeweb/articles/656947/> (дата обращения – 18.12.2023)
3. WebRTC For The Curious [Электронный ресурс]. URL: <https://webrtcforthecurious.com/> (дата обращения – 18.12.2023)