# Технология WebRTC

Для обмена медиа-данными, а именно видео, и аудио в работе использовалась технология *WebRTC*. *WebRTC* - (*Web Real-Time Communication*) -  открытая технология передачи потоковых данных в браузере по технологии точка-точка. На данный момент поддерживается большинством современных браузеров. Использование технологии позволяет отказаться от сторонних приложений для обмена потоками данных, например, от приложения скайп.

Перед началом использования технологии был проведен анализ преимуществ и недостатков технологий, а также была составлена таблица поддержи браузерами технологии *WebRTC* (табл.1).

Таблица 1- поддержка технологии WebRTC браузерами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Браузер** | **Поддержка WebRTC** | **Поддержка VP8** | **Поддержка H.264** | **Захват экрана** |
| *Google Chrome* | Да | Да | Да | Да |
| *Mozilla FireFox* | Да | Да | Да | Да |
| *Opera* | Да | Да | Да | Да |
| *Microsoft Edge* | Да | Нет | Да | Нет |
| *Safari* | Да | Нет | Да | Нет |
| *IE* | Нет | Нет | Нет | Нет |

Из таблицы совместимости браузеров (табл. 1) видно, что только *Internet Explorer* не поддерживает технологию полностью. Браузеры *Safari* и *Microsoft Edge* поддерживают технологию частично: поддержка *WebRTC* технологии и поддержка кодирования в *H.264* имеется в этих браузерах. Современные браузеры, такие *как google chrome, firefox и opera* поддерживают технологию полностью.

Из преимуществ технологии можно отметить следующее:

1. Не требует установки
2. Высокое качество связи
3. Защищенность соединения
4. Гибкость в реализации интерфейса
5. Открытый исходный код
6. Кроссплатформенность

А из недостатков:

1. Отсутствие стандарта сигнализации
2. Отсутствие возможности видеоконференций по умолчанию

Учитывая то небольшое количество недостатков, а также большое количество преимуществ технология была выбрана для организации видео-сообщения между узлами

**Использование и детали технологии**

Каждый клиент (браузер) поддерживающий технологию WebRTC имеет доступ к API (application programming interface) медиа объекта (MediaStream), в который необходимо указать тип и источник медиа данных. В свою очередь медиа поток состоит из каналов (MediaTrack объектов), где нужно cконфигурировать различные виды поток (аудио, видео).

Например, в случае данной работы, необходимо было создать на каждом клиенте один общий медиа поток состоящий из двух медиа-каналов - аудио и видео и синхронизировать их между собой. Для указания типов медиа-каналов нужно указать в конфигурации свойство *kind* (*video*, *audio*), а для синхронизации достаточно воспользоваться свойством *label* и установить его одинаковым для синхронизируемых каналов.

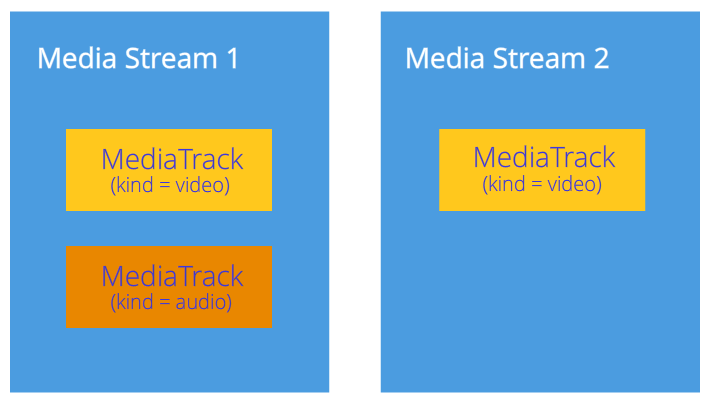


Рисунок 1 - конфигурация медиа потоков

Для создания соединения клиенты должны договориться о двух вещах:

1. О логическом соединении - какой формат данных, какие кодеки, тип данных
2. О физическом соединении - IP-адреса, порты, тип соединения (TCP, UDP)

# Логическое соединение - SDP

SDP - (Session Description Protocol) - протокол дескриптора сессии. Так как устройства на клиентах всегда будут разные - различные веб-камеры, микрофоны, различные кодеки и драйвера используемые на этих устройствах, то их необходимо каким-то образом скоординировать для взаимодействия. Для этого в *WebRTC* имеется дескриптор сессии - SDP. Дескриптор *SDP* передается от одного клиента другому с помощью сигнального механизма, речь о котором пойдет дальше. После получения вторым клиентом *SDP* дескриптора, он создаст свой дескриптор на основе совпадений конфигураций с полученным дескриптором от первого узла, который необходим будет отправить обратно.

Например, на рис… клиент 1 отправляет в дескрипторе информацию о поддержке кодеков А и B, клиент 2 поддерживает кодеки B и C. На основе этого клиент 2 решает, что информация для клиента 1 будет кодироваться кодеком B и сообщает ему об этом.  Таким образом клиенты обмениваются дескрипторами и приходят к общему соглашению о формате обмена данными.

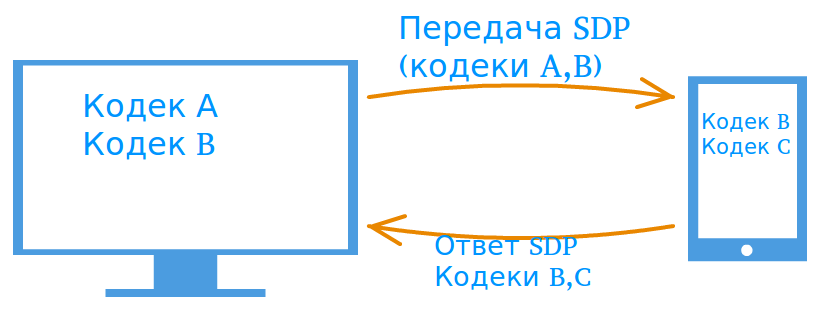


Рисунок 2 - обмен клиентов дескрипторами SDP

# Физическое соединение

Логическое соединение с помощью SDP дескриптором было установлено. Теперь необходимо установить физическое. Для этого используется ICE протокол (Ice candidate). Он содержит информацию о физическом расположении клиента и протоколе передачи информации по сети (IP, PORT, TCP/UDP). Также как и SDP генерируется в WebRTC и передается сигнальному серверу для дальнейшей отправки второму клиенту. Второй клиент, получив *Ice candidate* токены содержащие информацию о расположении будет знать с кем и по какому адресу устанавливать соединение.

# Сигнальный сервер

Сигнальный сервер, как описано выше, необходим для координации узлов, а именно для обмена данными о физическом расположении узлов в сети и их способам кодирования информации. В качестве сигнального сервера для координирования клиентов используется сторонний сервис *Twilio*. Сервис предлагает *STUN*-сервер, необходимый для физического соединения узлов, не имеющих публичных IP-адресов, а находящихся “за маршрутизаторами”. Другими словами, *STUN*-сервер делает доступным клиент за пределами сети.

Суммируя описанное выше процесс работы технологии *WebRTС* можно описать всего в основных 5 шагов:

1. Инициализируется медиа-объект *WebRTC* узлом один
2. Информация о инициализирующем узле (один) отправляется на сигнальный сервер
3. Сигнальный сервер отправляет информацию от узла один узлу два
4. Узел два обрабатывает информацию узла один и в соответствии с ней отправляет ответную информацию узлу один через сигнальный сервер.
5. Начинается процесс передачи закодированных данных между клиентами в обход сигнального сервера

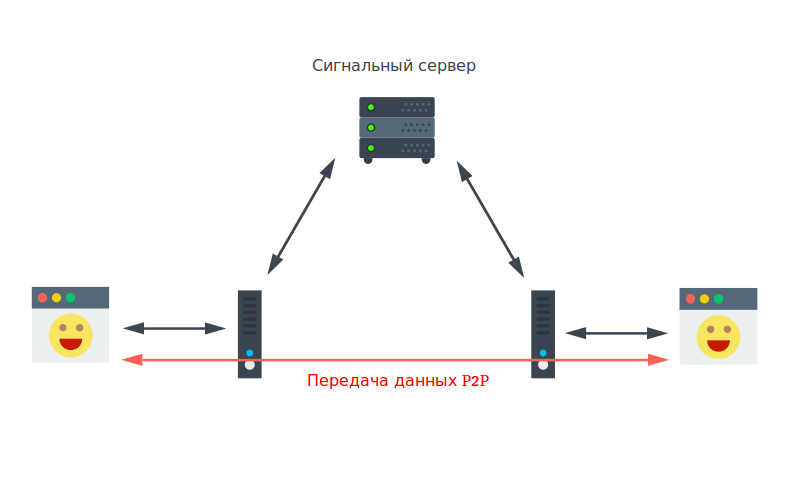


Рисунок 3 - схема работы WebRTC

Непосредственно для передачи медиа-потоков сигнальный сервер не нужен. Как видно из пунктов выше, сигнальный сервер нужен лишь для обмена метаданными для установления соединения типа точка-точка.