

ВВЕДЕНИЕ

WebRTC — технология, которая позволяет устанавливать соединение между двумя и более клиентами, через которое могут передаваться как потоковые видео и аудио данные, так и тестовые сообщения и медиаданные с помощью браузера.

С помощью технологии WebRTC компании имеют шанс трансформировать связь, предоставляя надежные и безопасные коммуникации корпоративного класса. Это открывает возможности для организации онлайн-совещаний, видеоконференций и других мероприятий.

Актуальность данной работы обусловлена тем, что в настоящее время онлайн-конференции на базе технологии WebRTC – это новый этап в развитии интернет-коммуникаций. Данный стандарт позволяет превратить браузер в оконечный терминал видеоконференцсвязи, что дает преимущество перед другими технологиями за счет того, что общение происходит в реальном времени без установки плагинов и иных расширений.

Целью работы является создание программного продукта стандарта для установления соединения и обмена информацией между конечными пользователями при проведении видеоконференции.

Для достижения поставленной цели были поставлены следующие задачи:

- Изучить основные теоретические сведения по стандарту webRTC;
- Разработка программного обеспечения для организации передачи потоковых данных между браузерами;
- Сравнение разработанного проекта с другими платформами вещания в реальном времени.

					ПП.920000.000	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

Объектом исследования является онлайн-конференция в режиме реального времени.

Предмет исследования — сервис для осуществления онлайн-конференции на базе технологии webRTC.

					ПП.920000.000	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1. Описание информационной системы

1.1 Описание стандарта WebRTC

WebRTC (Web Real Time Communications) — это стандарт, который описывает передачу потоковых аудиоданных, видеоданных и контента от браузера и к браузеру в режиме реального времени без установки плагинов или иных расширений. Стандарт позволяет превратить браузер в оконечный терминал видеоконференцсвязи, достаточно просто открыть веб-страницу, чтобы начать общение.

Передача данных осуществляется с помощью SRTP (Secure Real-time Transport Protocol) протокола. WebRTC использует DTLS-SRTP для шифрования, аутентификации и целостности сообщений, а также для защиты от атак повторного воспроизведения. Это дает конфиденциальность за счет шифрования RTP-нагрузки и проверки подлинности. SRTP это один из компонентов для безопасности, он очень удобен для разработчиков, которые ищут надежное и безопасное API.

SRTP (Secure Real-time Transport Protocol) это система безопасности, которая расширяет протокол RTP (Real-time Transport Protocol) набором защитных механизмов. Протокол был опубликован организацией IETF (Internet Engineering Task Force) в стандарте RFC 3711, в марте 2004.

SRTP обеспечивает конфиденциальность за счет шифрования RTP-нагрузки, не включая заголовки RTP. Также поддерживается проверка подлинности, которая широко используется как защитный механизм в RTP. SRTP можно использовать не с полным набором

В WebRTC используются два аудиокодека, G.711 и Opus, а также видеокодеки VP8 и H.264. В большинстве случаев разработчики предпочитают использовать Opus аудиокодек и VP8 видеокодек в силу их характеристик.

					ПП.920000.000	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

1.2 Описание работы

Установить соединение p2p – довольно трудная задача, так как компьютеры не всегда обладают публичными IP адресами, то есть адресами в интернете. Из-за небольшого количества IPv4 адресов (и для целей безопасности) был разработан механизм NAT, который позволяет создавать приватные сети, например, для домашнего использования. Многие домашние роутеры сейчас поддерживают NAT и благодаря этому все домашние устройства имеют выход в интернет, хотя провайдеры интернета обычно предоставляют один IP адрес. Публичные IP адреса - уникальны в интернете, а приватные нет. Поэтому соединиться p2p - трудно.

1.2.1 Преобразование сетевых адресов

Прежде чем говорить о P2P необходимо ознакомиться с тем, как общаются клиенты, находящиеся в разных сетях с помощью механизма NAT.

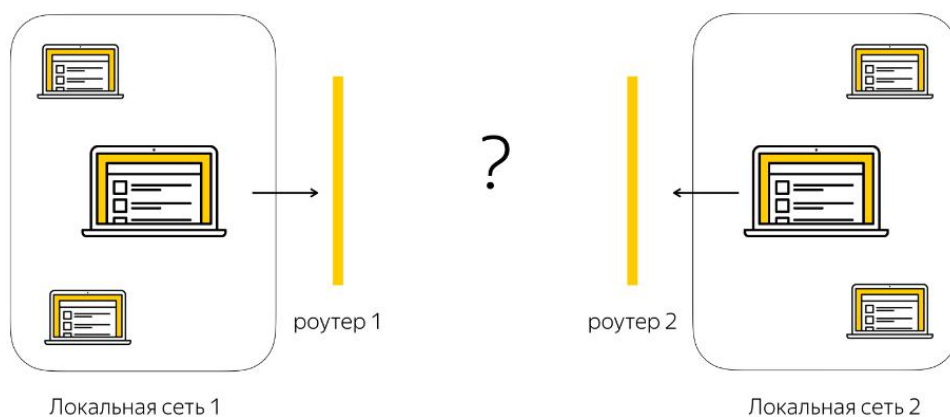


Рисунок - Работа NAT

В интернете многие компьютеры скрыты за роутерами. Есть локальные сети, внутри которых компьютеры знают свои адреса, есть роутер, который обладает внешним IP-адресом, и наружу все компьютеры внутренней сети

идентифицированы с IP-адресом этого роутера. Когда пакет от компьютера в локальной сети идет на роутер, роутер смотрит, куда его нужно переадресовать. Если пакет необходимо передать на другое устройство в этой же локальной сети, то он просто его ретранслирует, а если нужно его отправить вовне, в интернет, то составляется таблица маршрутизации.

внутренний IP	внутренний порт	внешний IP	внешний порт
192.168.0.15	35825	10.6.201.7	820

Рисунок - Таблица маршрутизации

В таблицу заполняется внутренний IP-адрес компьютера, который желает переслать пакет, его порт, ставится внешний IP-адрес то есть IP-адрес роутера, и делается подмена порта. Она нужна для того, чтобы правильно смаршрутизировать ответные пакеты. Мы их будем идентифицировать по порту, порт будет по каждому из компьютеров уникальным, в то время как внешний IP-адрес будет совпадать. В следующих главах будет рассмотрено, как пользователи могут передавать информацию в обход NAT.

1.2.2 Установка P2P соединения

Чтобы соединить два узла через с помощью WebRTC необходимо провести некие предварительные действия для установления соединения. Это первая фаза – установка соединения. Вторая фаза – передача видео-данных.

Сразу стоит сказать, что, хоть технология WebRTC в своей работе использует множество различных способов коммуникации и имеет гибкое переключение между ними, она не имеет протокола для передачи данных о

соединении. Не удивительно, ведь подключить два узла не так-то просто. Поэтому необходимо иметь некоторый дополнительный способ передачи данных, никак не связанный с WebRTC. Это может быть сокетная передача, протокол HTTP, это может быть даже протокол SMTP. Все данные передаются в виде текста и делятся на два типа – SDP и Ice Candidate.

Рассмотрим способ установки соединения между двумя клиентами. Первый клиент желает совершить звонок второму клиенту. WebRTC дает всю необходимую информацию, чтобы себя обозначить. Но остается открытым вопрос, как одному браузеру найти другой, как эту метайнформацию переслать, как проинициализировать вызов.



Рисунок - Установка P2P соединения

Предположим, что существует сигнальный сервер, который держит постоянное соединение с нашими клиентами, например, по веб-сокетам или с помощью HTTP. Стандарт wtrTC так же не описывает никаких требований для такого сервера, поэтому разработчик может как выбрать общедоступный сервер, так и поднять свой. Первый клиент формирует метайнформацию, и с помощью веб-сокетов или HTTP пересылает ее на сигнальный сервер. Пересылает также какую-то часть информации, с кем именно он хочет соединиться, например, никнейм или еще какую-то информацию.

Сигнальный сервер по этому идентификатору устанавливает, какому именно клиенту нужно переадресовать нашу метаинформацию, и пересылает ее. Второй клиент берет ее, использует, устанавливает себе, формирует ответ, и с помощью сигнального механизма пересылает ее на сигнальный сервер, тот в свою очередь ретранслирует ее первому клиенту. Таким образом оба клиента в данный момент обладают всей необходимой датой и метаинформацией, чтобы установить P2P-соединение.

По сути клиенты обмениваются обычным текстом. Он содержит в себе информацию об IP-адресе, о портах, которые используются, о том, какая именно информация гоняется между клиентами, что это такое — аудио, видео, какие кодеки используются. Все, что нам необходимо, там есть.

```
v=0
o=- 1815849 0 IN IP4 192.168.0.15
s=Cisco SDP 0
c=IN IP4 194.67.15.181
t=0 0
m=audio 20062 RTP/AVP 99 18 101 100
a=rtpmap:99 G.729b/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=rtpmap:100 X-NSE/8000
a=fmtp:100 200-202
```

Рисунок - SDP датаграмма

Следует обратить внимание на вторую строчку. Там указан IP-адрес клиента, 192.168.0.15. Очевидно, это IP-адрес компьютера, который находится в какой-то локальной сети. Но, для того, чтобы клиенты смогли установить соединение, они должны знать внешние IP-адреса участников

системы. И в этом помогает протокол ICE — Internet Connectivity Establishment, с помощью которого можно будет получить внешние адреса.

1.2.3 STUN и TURN протоколы

Существуют протоколы, использующие пакеты UDP для передачи голоса или изображений по IP-сетям. Если обе общающиеся стороны находятся за NAT'ом, соединение не может быть установлено обычным способом. Именно здесь STUN и оказывается полезным.

STUN — это клиент-серверный протокол. Клиент может включать в себя реализацию клиента STUN, который отправляет запрос серверу STUN. Затем сервер STUN отправляет клиенту обратно информацию о том, каков внешний адрес маршрутизатора NAT, и какой порт открыт на NAT для приема входящих запросов обратно во внутреннюю сеть.

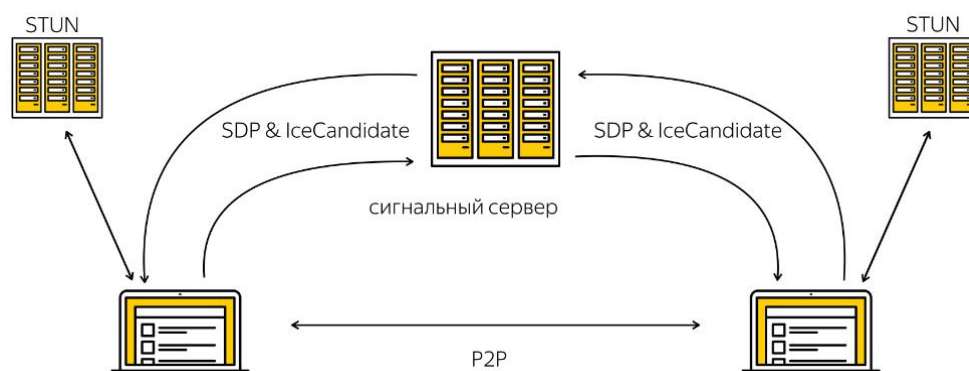


Рисунок - Получение внешних IP-адресов с помощью STUN

Таким образом, в процессе установки P2P соединения, каждый из клиентов должен сделать по запросу к этому STUN-серверу, чтобы узнать свой IP-адрес, и сформировать дополнительную информацию, IceCandidate, и с помощью сигнального механизма также этим IceCandidate обменяться. Тогда клиенты будут знать друг о друге с правильными IP-адресами, и смогут установить P2P соединение.

STUN предусматривает одно средство для прохождения NAT и позволяет клиенту получить транспортный адрес (IP адрес и порт), который может быть полезен для приема пакетов от реер-ов. Однако адреса, полученные через STUN, не могут быть доступны всем реер-ам. Эти адреса работают в зависимости от топологии сети. Таким образом, STUN сам по себе не может обеспечить комплексное решение для обхода NAT.

Так же STUN не подходит, когда компьютер скрыт за двойным NAT. В этом случае фреймворк ICE предписывает использование TURN-сервера.

Traversal Using Relay NAT (TURN) — это протокол, который позволяет узлу за NAT или брандмауэром получать входящие данные через TCP или UDP соединения. Такая возможность особенно актуальна для узлов позади симметричных NAT, или брандмауэров, которые собираются стать принимающей стороной в соединении с одним конкретным узлом (реер-ом).

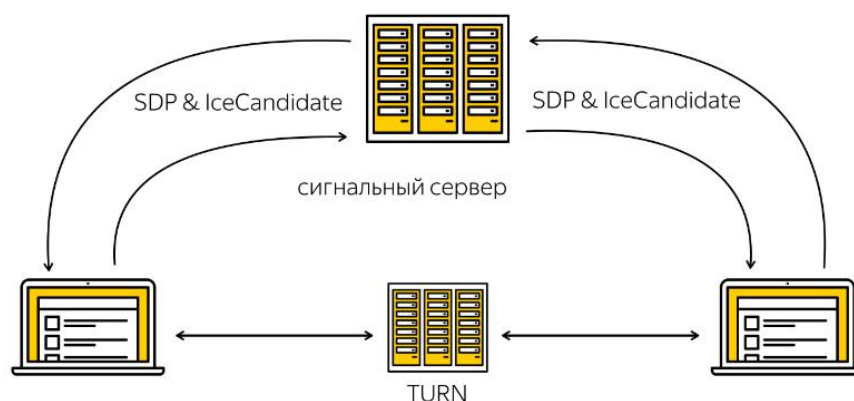


Рисунок - Обмен информации через TURN

В данном случае TURN выступает специальным сервером, который превращает соединение клиент-клиент, P2P, в соединение клиент-сервер-клиент, то есть выступает в роли ретранслятора. Независимо от того, по какому из трех сценариев пошла установка соединения, находятся ли клиенты в локальной сети, нужно ли обратиться к STUN- или TURN-серверу, API-технология остается неизменной.

TURN будет почти всегда обеспечивать подключение к клиенту, но в то же время он создает большую нагрузку на провайдера TURN-сервера. Поэтому рекомендуется использовать TURN только в крайнем случае, предпочитая другие механизмы (например, STUN или прямое подключение), когда это возможно. Для достижения этого может использоваться методология Interactive Connectivity Establishment (ICE), чтобы найти оптимальное средство связи.

1.2.4 Варианты работы webRTC

WebRTC осуществляет соединение между конечными пользователями, позволяя осуществлять обмен потоковыми данными без участия сервера. Но, если требуется участие более двух пользователей в системе, то без сервера уже не обойтись, так как нецелесообразно и слишком затратно устанавливать оконченное соединение между всеми пользователями.

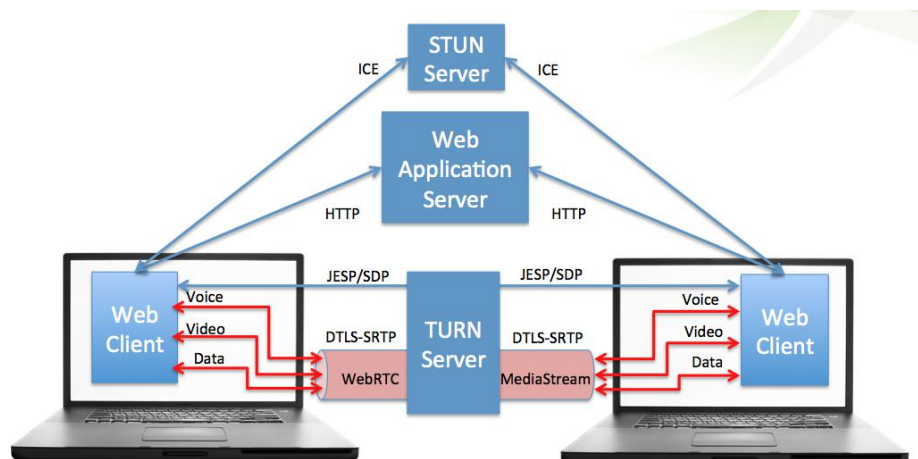


Рисунок 2 - Схема работы webRTC между клиентами

Для обмена данными между несколькими участниками необходимо установить передачу данных через сервер. Он необходим для получения трафика со всех участников системы для обработки и преобразования в

удобный для пользователей вид. Также WebRTC сервер будет получать медиа-трафик от WebRTC пиров и передавать его участникам конференции, которые используют приложения для настольных компьютеров или мобильных устройств, в случае наличия таковых.

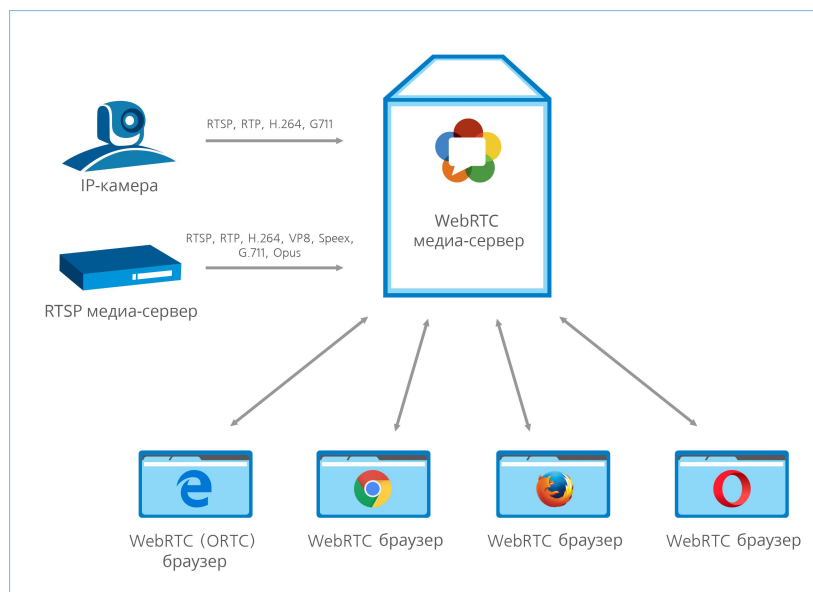


Рисунок - Схема работы webRTC с помощью сервера

1.3 Безопасность

Шифрование и безопасность являются встроенными средствами по умолчанию. Более того, на большинстве серверов WebRTC предоставляет сквозное шифрование между узлами, тем самым обеспечивая безопасную передачу данных в режиме реального времени.

При взаимодействии с сайтом по протоколу HTTPS веб-браузер гарантирует, что пользователь обращается именно к выбранному сайту, а также защищен от любых сетевых атак. Сервисам обмена сообщениями, электронной почты и другим, использующим сайты как посредника, для защиты нужны дополнительные средства, такие как WebRTC.

В большинстве веб-приложений безопасность предоставляется на уровне двухточечного соединения. Пользователь хочет произвести соединение с сайтом <https://example.com>, браузер начинает TLS-сеанс с

сервером, который объявляет себя example.com, и при помощи полученных им данных удостоверяется в том, что соединение правильное. В то время, как TLS обеспечивает целостность данных и конфиденциальность, а браузер выполняет аутентификацию, сверяя пароль, с помощью которого провайдер идентификации предоставляет подтверждение о связи между открытым ключом сервера и его «личностью». Опознанная «личность» считается логическим источником данных и необходимого для его инициализации кода.

Использование для сигнализации протокола HTTPS позволяет защититься от сетевых атак между двумя браузерами и сервером.

WebRTC использует протокол датаграмм безопасности транспортного уровня для передачи данных в режиме реального времени — DTLS (Datagram Transport Layer Security). Данный протокол встроен во все браузеры, которые поддерживают технологию WebRTC, такие как Chrome, Firefox и Opera. В связи, которая зашифрована при помощи DTLS, исключается подделка информации и подслушивание.

Помимо DTLS, технология WebRTC применяет для шифрования медиаданных протокол передачи данных SRTP (Secure Real-Time Protocol). Данный протокол исключает просмотр или прослушивание IP-связи третьими лицами.

Технология не требует установки ничего дополнительного, никаких плагинов, и это хорошо. И не получится сделать шпионское приложение, сайт не будет подслушивать или подсматривать за пользователем, он покажет пользователю специальный промт, запросит у него доступ и получит его, только если пользователь разрешит доступ к аудио- и медиаустройствам.

1.4 Аудио- и видеоинструменты в WebRTC

					ПП.920000.000	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

Богатый опыт проведения телеконференций в браузере требует, чтобы браузер имел доступ к системному оборудованию для захвата аудио и видео. Никаких сторонних плагинов или настраиваемых драйверов, только простой и последовательный API. Однако сырые аудио- и видеопотоки также недостаточны сами по себе: каждый поток должен обрабатываться для повышения качества, синхронизации, а выходной битрейт должен регулироваться непрерывно изменяющейся полосой пропускания и задержкой между клиентами.

На принимающей стороне процесс изменяется. Клиент должен декодировать потоки в режиме реального времени и иметь возможность настраиваться на задержки сети. Можно сделать вывод что, захват и обработка аудио и видео - сложная проблема. Однако WebRTC предоставляет браузеру полнофункциональные аудио- и видеосистемы, которые заботятся обо всей обработке сигналов.

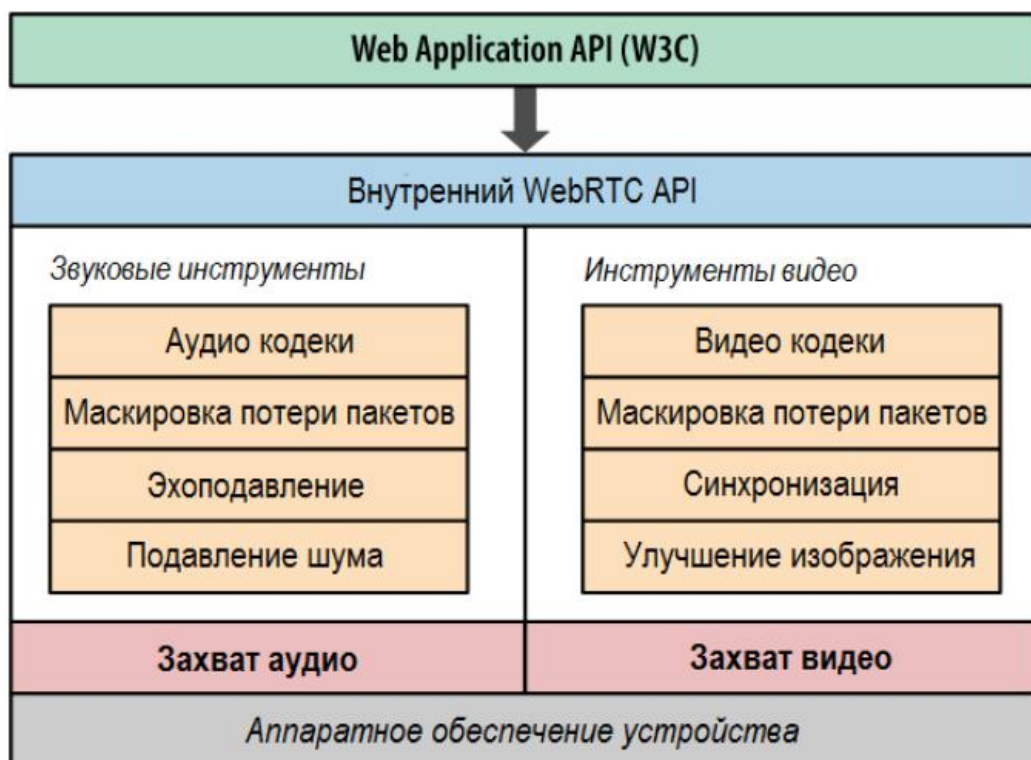


Рисунок - Функциональность аудио- и видеосистем

Полученный аудиопоток обрабатывается для уменьшения шума и эхоподавления, затем автоматически кодируется одним из оптимизированных узкополосных или широкополосных аудиокодеков. Наконец, специальный алгоритм сокрытия ошибок используется, чтобы скрыть негативные последствия потери пакетов. Видеопроцессор выполняет аналогичную обработку, оптимизируя качество изображения, подбирая оптимальные параметры сжатия и настройки кодека, применяя маскировку потери пакетов и многое другое.

Вся обработка выполняется непосредственно браузером, и что еще более важно, браузер динамически корректирует свой конвейер обработки, чтобы учитывать постоянно изменяющиеся параметры аудио- и видеопотоков и сетевых условий. После того как вся эта работа будет завершена, веб-приложение получает оптимизированный поток, который затем может выводиться на локальный экран и динамики.

1.5 Преимущества и недостатки стандарта

У любого стандарта и технологии есть как положительные, так и отрицательные стороны. Преимуществами данного являются:

- Не требуется установка ПО - соединение может осуществляться прямо в браузере;
- Высокое качество связи - осуществляется благодаря использованию современных видео (VP8, H.264) и аудио (Opus) кодеков, автоматического подстраивания качества потока под условия соединения. Так же такое качество связи обеспечивается посредством встроенной системы эхо- и шумоподавления, а так же автоматической регулировки уровня чувствительности микрофонов участников;
- Высокий уровень безопасности - все соединения защищены и зашифрованы на протокольном уровне с помощью SSL/TLS и SRTP;

- Есть встроенный механизм захвата контента - например захват рабочего стола;
- Возможность реализации любого интерфейса управления - интерфейс можно реализовать с помощью языка разметки HTML5, таблиц стилей CSS и языка программирования JavaScript;
- Возможность интеграции интерфейса с любыми back-end системами с помощью WebSockets.
- Проект с открытым исходным кодом - есть возможность для некоммерческого использования и внедрения в программные продукты;
- Настоящая кросс-платформенность - пользователи любых устройств, операционных систем и браузеров могут установить соединение между собой при наличии интернет соединения.

Недостатками стандарта являются:

- Для организации групповых аудио и видеоконференций требуется сервер ВКС, который бы микшировал видео и звук от участников, т.к. браузер не умеет синхронизировать несколько входящих потоков между собой.
- Все WebRTC решения несовместимы между собой, т.к. стандарт описывает лишь способы передачи видео и звука, оставляя реализацию способов адресации абонентов, отслеживания их доступности, обмена сообщениями и файлами, планирования и прочего за вендором.
- Другими словами вы не сможете позвонить из WebRTC приложения одного разработчика в WebRTC приложение другого разработчика.
- Микширование групповых конференций требует больших вычислительных ресурсов, поэтому такой тип видеосвязи требует покупки платной подписки либо инвестирования в свою инфраструктуру, где на каждую конференцию требуется 1 физическое ядро современного процессора.

1.6 Поддержка браузерами

Стандарт-технология WebRTC поддерживается почти всеми популярными браузерами, но есть и исключения. Политика Microsoft такова, что компания не приветствует внедрение технологии, которые они не контролируют. Но постепенно дело сдвинулось с мёртвой точки, т.к. игнорировать WebRTC далее было уже нельзя, и был анонсирован проект ORTC, производный от стандарта WebRTC.

По словам разработчиков ORTC — это расширение стандарта WebRTC с улучшенным набором API на основе JavaScript и HTML5, что в переводе на обычный язык означает, что всё будет то же самое, только контролировать стандарт и его развитие будет Microsoft, а не Google. Набор кодеков расширен поддержкой H.264 и некоторым аудиокодеками серии G.7XX, используемыми в телефонии и аппаратных ВКС системах. Возможно появится встроенная поддержка RDP (для передачи контента) и обмена сообщениями.

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			62		10.2				
		57	63		10.3				4
11	16	58	64	11	11.2	all	64	11.8	6.2
	17	59	65	11.1	11.3				
		60	66	TP					
		61	67						

Рисунок - Список браузеров, которые поддерживают webRTC

Но несмотря на это браузеры из компании Microsoft не пользуются большой популярностью на сегодняшний день, поэтому данными проблемами поддержки можно пренебречь.

2. Технический обзор и анализ актуальности данного протокола.

2.1 Медиа потоки

Основной сущностью является медиа поток, то есть поток видео и аудио данных, картинка и звук. Медиа потоки бывают двух типов – локальные и удаленные. Локальный получает данные от устройств входа (камера, микрофон), а удаленный по сети. Таким образом у каждого узла есть и локальный, и удаленный поток.



Рисунок - Схематическое изображение медиапотоков

В WebRTC существует определенная иерархия внутри потока. Каждый поток может состоять из нескольких медиа дорожек (MediaTrack), которые в

свою очередь могут состоять из нескольких медиа каналов (MediaChannel). Да и самих медиа потоков может быть тоже несколько.

Медиапотоки — это такие каналы, которые внутри себя содержат треки. Треки внутри медиа-потока синхронизированы. Аудио и видео не будет расходиться, они будут идти с единым таймингом. Вы можете внутри медиапотока сделать любое количество треков, треками можно управлять по отдельности, например, вы можете приглушить аудио, оставив только картинку. Также вы можете передавать любое количество медиа-потоков, что позволяет вам, например, реализовать конференцию.

Должно быть понятно, что медиа поток как минимум должен включать в себя возможность содержать данные разных типов — видео и аудио. Это учтено в технологии и поэтому каждый тип данных реализуется через медиа дорожку MediaTrack. У медиа дорожки есть специальное свойство kind, которое и определяет, что перед нами – видео или аудио.

В случаях, если необходимо передать два медиа потока, когда необходимо передать поток с веб-камеры и трансляцию рабочего стола, то на медиа потоки устанавливается свойство label – метка потока, его название. Но если медиа дорожки можно идентифицировать через метку, то зачем нам для использовать два медиа потока, вместо одного? Ответ кроется в важнейшем свойстве медиа потоков – они синхронизируют медиа дорожки. Разные медиа потоки не синхронизируются между собой, но внутри каждого медиа потока все дорожки воспроизводятся одновременно.

Таким образом, если требуется, чтобы видео с веб-камеры и трансляция экрана воспроизводились одновременно, и, если медиа потоки зависимы друг от друга, то стоит их объединить и использовать один медиа поток. Если это не столь важно, то выгодней использовать разные потоки, так как картинка будет более гладкой.

Для передачи звука так же используются отдельные каналы. В случаях, если необходимо передавать стерео звук, то нужно помнить, что стерео звук

					ПП.920000.000	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

– это два разных звука. И передавать их надо тоже отдельно. Для этого используются каналы MediaChannel. Медиа дорожка звука может иметь много каналов (например, 6, если нужен звук 5+1). Внутри медиа дорожки каналы, разумеется тоже синхронизированы. Для видео обычно используется только один канал, но могут использоваться и несколько, например, для наложения рекламы.

В самой же простой ситуации для реализации видеочата потребуется организовать один медиа поток, который будет состоять из двух дорожек - видео дорожки и аудио дорожки, каждая из которых будет состоять из одного основного канала. Видео дорожка отвечает за камеру, аудио дорожка – за микрофон, а медиа поток – это контейнер их обоих.

2.2 Обзор API для установления соединения

Стандарт-технология WebRTC базируется на трех основных API (Application programming interface - программный интерфейс приложения):

- **MediaStream** - отвечает за принятие веб-браузером аудио и видеосигнала от камер или рабочего стола пользователя;
- **RTCPeerConnection** - отвечает за соединение между браузерами для “обмена” полученными от камеры, микрофона и рабочего стола, медиаданными. Также в “обязанности” этого API входит обработка сигнала (очистка его от посторонних шумов, регулировка громкости микрофона) и контроль над используемыми аудио и видеокодеками;
- **RTCData Channel** - обеспечивает двустороннюю передачу данных через установленное соединение.

MediaStream. Прежде чем получить доступ к микрофону и камере пользователя, браузер запрашивает на это разрешение. В Google Chrome можно заранее настроить доступ в разделе “Настройки”, в Opera и Firefox выбор устройств осуществляется непосредственно в момент получения доступа, из выпадающего списка. Запрос на разрешение будет появляться

всегда при использовании протокола HTTP и однократно, если использовать HTTPS.

RTCPeerConnection. Каждый браузер, участвующий в WebRTC конференции, должен иметь доступ к данному объекту. Благодаря использованию RTCPeerConnection медиаданные от одного браузера к другому могут проходить даже через NAT и сетевые экраны. Для успешной передачи медиапотокa участники должны обмениваться следующими данными с помощью транспорта, например, веб-сокета:

- участник-инициатор направляет второму участнику Offer-SDP (структура данных, с характеристиками медиапотока, которые он будет передавать);
- второй участник формирует “ответ” — Answer-SDP и пересылает его инициатору;
- затем между участниками организуется обмен ICE-кандидатами, если таковые обнаружены (если участники находятся за NAT или сетевыми экранами).

После успешного завершения данного обмена между участниками организуется непосредственно передача медиапотокa (аудио и видео).

RTCDatа Channel. Поддержка протокола Data Channel появилась в браузерах сравнительно недавно, поэтому данный API можно рассматривать исключительно в случаях использования WebRTC в браузерах Mozilla Firefox 22+ и Google Chrome 26+. С его помощью участники могут обмениваться текстовыми сообщениями в браузере.

2.3 Получение медиа потока

Интерфейс Navigator представляет собой состояние и особенности(свойства) пользовательского агента. Это позволяет скриптам

узнавать их и самостоятельно регистрироваться для выполнения некоторых действий.

Существует метод `getUserMedia`, который принимает на вход набор констрейнтов. Это специальный объект, где клиент указывает, к каким именно устройствам он хочет получить доступ, к какой именно камере, к какому микрофону. Указываются характеристики, которые хочет иметь клиент, какое именно разрешение, и есть также два аргумента — `successCallback` и `errorCallback`, который вызывается в случае успеха или неудачи. В более современных реализациях технологии используются промисы.

Есть также удобный метод `enumerateDevices`, который возвращает список всех подключенных к вашему компьютеру медиа-устройств, что дает вам возможность показать их пользователю, нарисовать какой-то селектор, чтобы пользователь выбрал, какую конкретно камеру он хочет использовать.

```
navigator.mediaDevices.getUserMedia(constraints, successCallback, errorCallback);
```

```
navigator.mediaDevices.enumerateDevices();
```

Рисунок - Получение медиа потока

Центральным объектом в API служит `RTCPeerConnection`. Когда выполняется соединение, то берем класс `RTCPeerConnection`, который возвращает объект `peerConnection`. В качестве конфигурации указывается набор ICE-серверов, то есть STUN- и TURN-серверов, к которым клиент будет обращаться в процессе установки. И есть важный ивент `onicescandidate`, который триггерится каждый раз, когда клиенту нужна помощь сигнального механизма. То есть технология WebRTC сделала запрос, например, к STUN-серверу, клиент узнал свой внешний IP-адрес, появился новый

					ПП.920000.000	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

сформированный ICECandidate, и нужно переслать его с помощью стороннего механизма, ивент стриггерился.

```
const peerConnection = new RTCPeerConnection({
  iceServers: [
    {
      urls: "turn:" + turnHost,
      username: "webrtc",
      credential: "turnserver"
    },
    ...
  ]
});

peerConnection.onicecandidate = function (evt) { };
```

Рисунок - Создание p2p соединения

Когда устанавливается соединение и требуется проинициализировать вызов, то используется метод createOffer(), чтобы сформировать начальную SDP, offer SDP, ту самую мета-информацию, которую нужно переслать другому участнику системы.

Чтобы установить ее в PeerConnection, используется метод setLocalDescription(). Собеседник получает эту информацию по сигнальному механизму, устанавливает ее себе с помощью метода setRemoteDescription() и формирует ответ с помощью метода createAnswer(), который также с помощью сигнального механизма пересылается первому клиенту.

```

peerConnection.createOffer();

peerConnection.setLocalDescription();

peerConnection.setRemoteDescription();

peerConnection.createAnswer();

```

Рисунок - Создание и установка SDP

Когда клиент получил доступ к медиа, медиапоток, он может его передать в уже имеющееся P2P-соединение с помощью метода `addStream`, а другой клиент узнает об этом, у него стриггерится ивент `onaddstream`. Он получит наш поток и сможет его отобразить.

```

peerConnection.addStream(stream);

peerConnection.onaddstream = function (evt) { };

```

Рисунок - Работа с потоками

Также API предоставляет доступ работать с дата-потоками. Очень похоже на формирование обычного `peerConnection`. Для этого нужно указать `RtpDataChannels: true` и вызвать метод `createDataChannel()`.

```

const peerConnection = new RTCPeerConnection(null, {
  optional: {
    RtpDataChannels: true
  }
});

const dataChannel = peerConnection.createDataChannel();

```

Рисунок - Работа с дата потоками

2.3 Практическое применение webRTC

Для голосовой и видеосвязи, а также для организации рабочих мест более не требуется установка каких-либо программ, приложений или «плагинов»: модуль WebRTC изначально встроен в современные браузеры и позволяет посетителю сайта SIPNET звонить на любые номера нажатием одной кнопки.

WebRTC-приложение SIPNET по сути является обычной интерактивной web-страницей и может исполняться на персональных компьютерах, планшетах и смартфонах вне зависимости от аппаратной платформы или операционной системы.

Сегодня WebRTC в полном объеме поддерживают Google Chrome, Mozilla Firefox, а также все продукты на базе Chromium (Опера, Яндекс.Браузер и другие).

Для звонков из браузера никакого дополнительного оборудования не нужно. При разговоре используются встроенный микрофон и динамики смартфона, планшета, ноутбука или компьютера. Если в устройстве нет микрофона или динамиков, можно использовать микрофон USB-камеры, Bluetooth-колонки и любую телефонную гарнитуру.

Первым сервисом, который запустил в коммерческую эксплуатацию звонки с веб-браузера, стал сервис Callbacker [9], который интегрировал в свой личный кабинет веб-телефон, на базе открытого продукта sipml5 [10].

Что было сделано:

Интеграция в личный кабинет sipml5 и его кастомизация с отменой дополнительных функций, которые не работают или работают, но нестабильно, в режиме экспериментального тестирования;

установка и настройка пропатченного Asterisk 11 ревизии 373330. Asterisk в последнем релизе 11 поддерживает WebSocket и транспорт SAVPF, что делает его совместимым с веб-телефоном sipml5. Необходима поддержка SRTP, поэтому собираем Asterisk с обязательными параметрами: --with-crypto --with-ssl --with-srtp. Настройка Asterisk ничем не отличается от обычной, за исключением настроек http для работы WebSocket.

2.4 Выбор webRTC как одно из основных направлений для компаний

Многие компании, которые относятся как к крупному бизнесу, так и к малому, сторонники свободного ПО делают свой выбор в пользу webRTC по нескольким причинам:

- Бесплатная связь с консультантами, менеджерами, службой поддержки клиентов;
- Организация горячих линий связи;
- Новые возможности для разработчиков приложений типа on-line консультант;
- Эффективная работа менеджера с CRM-системой;
- Соблюдение конфиденциальности — это новые клиенты. в необычной обстановке;

					ПП.920000.000	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

3. Разработка программного продукта

Для разработки программного продукта, который реализует установление видео связи между конечными пользователями был выбран язык программирования JavaScript.

В качестве фреймворка для создания интерфейсов был выбран мощный инструмент - vue.js, который позволяет при разработке акцентировать внимание только на архитектуре приложения и ее логике, не беспокоясь о том, как в приложении будет производиться работа с состоянием.

3.1 Структура программной реализации

Основной целью реализации данного программного средства было создание защищенного видеочата на основе стандарта webRTC. Программная реализация является веб-приложением, находящимся в общем доступе и состоит из двух частей представления: формы и видео контейнера.

Условно программа состоит из ряда последовательных действий - отправка сообщения на сигнальный и STUN сервера для получения необходимой информации для осуществления видео звонка и цикла, который с помощью открытого сокет-соединения слушает, передает и воспроизводит аудио и медиа потоки между участниками системы. Схема работы программы представлена на рисунке

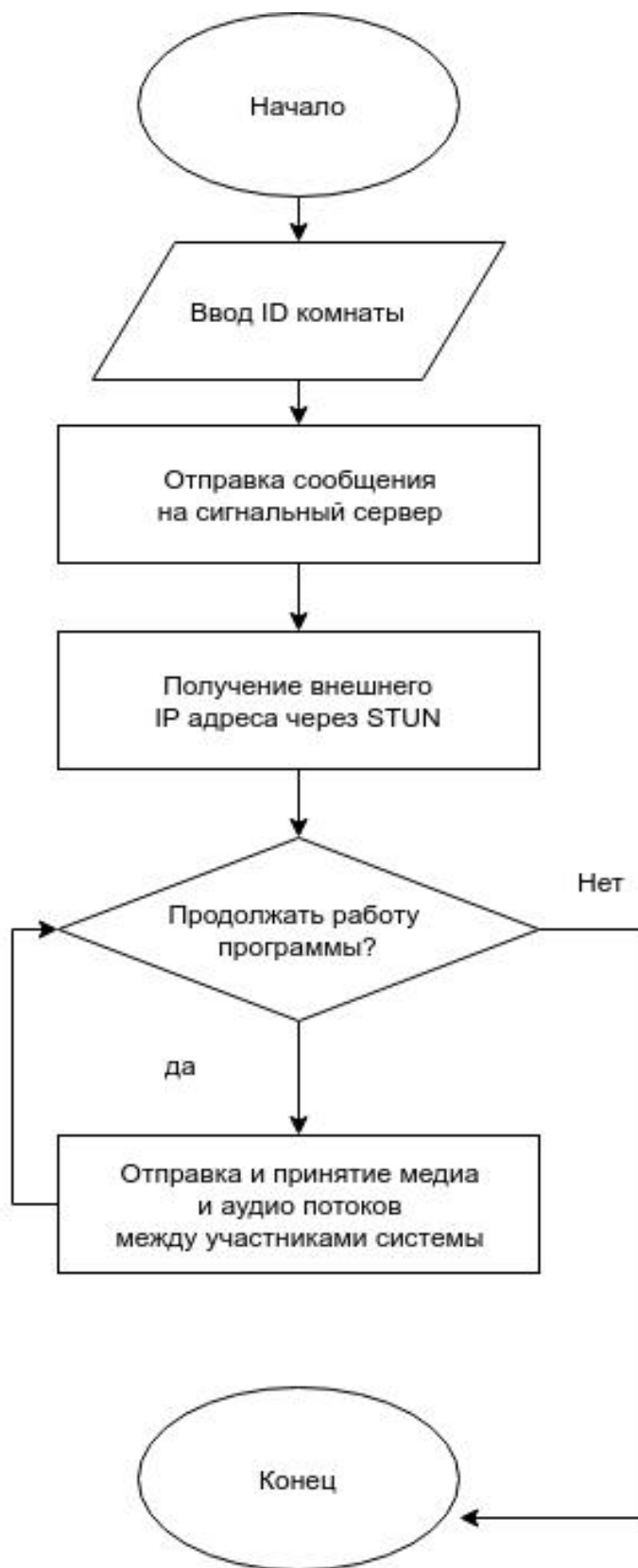


Рисунок - Блок-схема работы программы

3.2 Демонстрация работы программы

Язык гипертекстовой разметки HTML и таблицы стилей CSS предоставляют широкие возможности для создания визуальных форм.

Для пользования приложением пользователям не нужно устанавливать каких-либо плагинов или инструментов. Поэтому для того, чтобы начать пользоваться приложением - необходимо указать идентификатор комнаты, поле для ввода которого предоставлено интерфейсом программы

Очевидно, что интерфейс программы не будет выглядеть перегруженным и интуитивно понятен пользователю, который знает, как и зачем программа работает. На рисунке показана пользовательская форма программного решения.

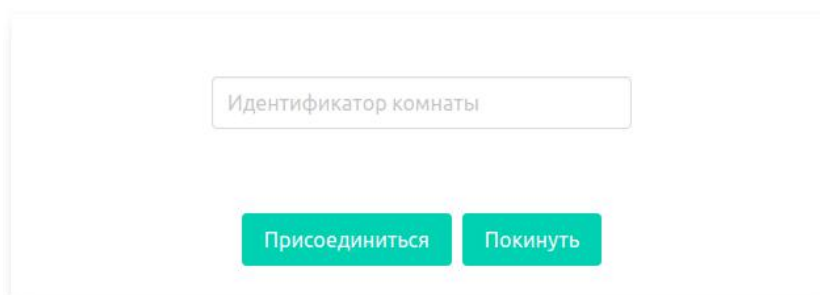
The image shows a simple web interface for a chat application. It features a light gray rectangular container. Inside the container, at the top, is a text input field with a light gray border and the placeholder text 'Идентификатор комнаты' in a small, gray font. Below the input field, centered horizontally, are two rectangular buttons. The left button is teal with the white text 'Присоединиться'. The right button is also teal with the white text 'Покинуть'.

Рисунок - интерфейс программы

Интерфейс представляет из себя поле для ввода идентификатора комнаты и две кнопки, которые позволяют либо присоединиться к комнате, либо ее покинуть. Идентификатором комнаты может являться любое значение строкового типа, которое заранее выбрали пользователи.

После ввода идентификационного номера участники системы нажимают кнопку «присоединиться», после чего соединение становится открытым. Для того, чтобы установилось соединение типа p2p необходимо дождаться, чтобы к комнате присоединилось два и более участников.

Для того, чтобы продемонстрировать работу приложения и создания конференции, в качестве клиентских терминалов будут выбраны компьютер, интернет-соединение которому предоставляет один провайдер, и мобильный телефон, интернет-соединение которому предоставляется другим провайдером. Этим самым можно повторить тот случай, когда участники видеоконференции находятся в разных сетях.

С компьютера необходимо перейти по ссылке приложения. Для присоединения к звонку нужно всего лишь ввести идентификационный номер комнаты и нажать кнопку «присоединиться».

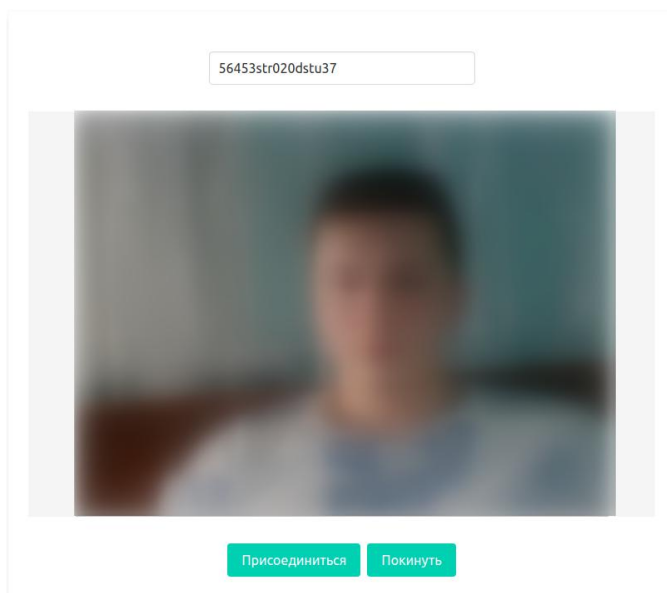


Рисунок - подключение одного участника системы

После того, как первый участник присоединился к комнате - система находится в состоянии ожидания.

Со второго устройства так же требуется перейти по ссылке и указать прежний идентификационный номер комнаты и присоединиться к комнате. После того, как второй пользователь присоединился к комнате - в окне приложения первого пользователя появляется новый контейнер, в который передается аудио и видео поток другого пользователя.

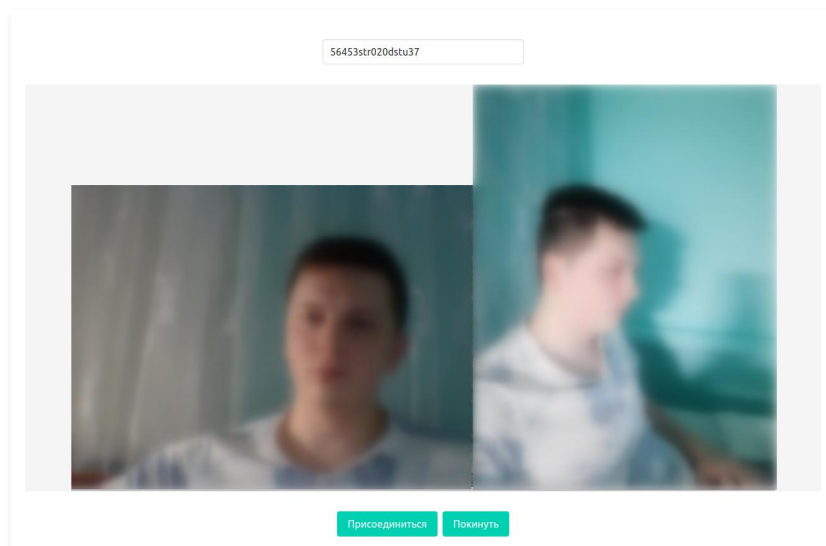


Рисунок - видео конференция между двумя участниками

После того, как второй пользователь присоединился на экране мобильного устройства так же отображается еще один контейнер, который содержит в себе потоковую передачу видео и аудио данных.

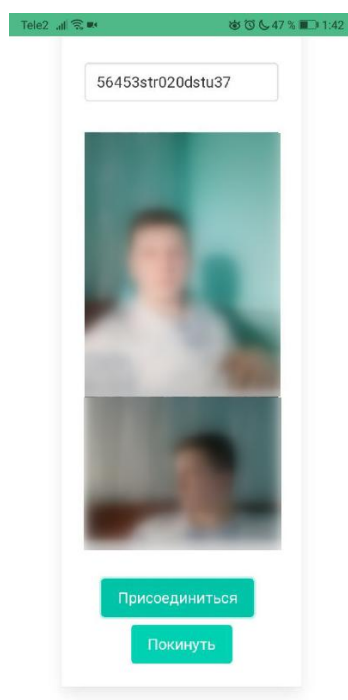


Рисунок - видео конференция между двумя участниками

ЗАКЛЮЧЕНИЕ

В ходе осуществления производственной практики были изучены основные теоретические сведения, касающиеся одной из новых технологий, которая позволяет обмениваться потоками данных между конечными пользователями.

В заключении данной работы стоит отметить, что был достигнут и выполнен ряд задач:

- изучены теоретические сведения по стандарту webRTC;
- изучены основы работы с webRTC с помощью API.
- определена актуальность использования стандарта webRTC.

					ПП.920000.000	Лист
Изм.	Лист	№ докум.	Подпись	Дата		36

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информационные системы и технологии / Под ред. Тельнова Ю.Ф.. - М.: Юнити, 2017. - 544 с.
2. Модель угроз и нарушителя безопасности персональных данных, обрабатываемых в специ-альных информационных системах персональных данных отрасли. Министерство связи и массовых коммуникаций Российской Федерации. Москва, 2010 [Электронный ресурс]. – Режим доступа: <http://minsvyaz.ru/common/upload/publication/1410084of.pdf>
3. IT Baseline protection manual. Standard BSI Электронный ресурс. 2004. - Режим доступа: <http://www.bsi.de/english/publications/index.htm>.
4. Варфоломеева, А.О. Информационные системы предприятия: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. - М.: НИЦ ИНФРА-М, 2017. - 283 с.
5. Галатенко В.А. Основы информационной безопасности. - ИНТУИТ. РУ "Интернет-университет Информационных Технологий", 2006. - 208 с.
6. Стивен Норткат, Джуди Новак. Обнаружение нарушений безопасности в сетях. Третье издание. Перевод с английского: Издательский дом «Вильямс», 2003 –448 стр. Стр. 75-87.
7. И. Д. Медведевский, П. В. Семьянов, Д. Г. Леонов «Атака на Internet». Издательство ДМК. -1999. -332стр. 120-125с.
8. Скудис Э. Противостояние хакерам. М.: ДМК Пресс, 2003. —506 с. – с. 349-370.
9. Callbacker - <http://www.callbacker.com/ru/index.html>
10. Simpl5 - <http://www.sipml5.org>