ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ/INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.23670/IRJ.2024.140.12

МЕТОДЫ РЕАЛИЗАЦИИ ВИДЕОСВЯЗИ В БРАУЗЕРЕ С ИСПОЛЬЗОВАНИЕМ OPEN-SOURCE ИНСТРУМЕНТОВ

Обзор

Кузьминов И.В.[1, *]

[1] Санкт-Петербургский государственный университет промышленных технологий и дизайна, Санкт-Петербург, Российская Федерация

* Корреспондирующий автор (ikuz.23[at]yandex.ru)

Аннотация

Общение в сети Интернет давно вышло на новый уровень. Мессенджеры и чаты для обмена короткими текстовыми сообщениями со смайликами сменились возможностью осуществления звонков в формате аудио и видео. И заслуга здесь в первую очередь в развитии технологий – ведь именно появление технологии WebRTC существенно упростило реализацию подобного рода связи для разработчиков. Технология бесплатна и проста в реализации, за счет чего она быстро получила широкое распространение. В рамках данной статьи будет выполнен обзор возможностей применения open-source решений для реализации видеосвязи в режиме реального времени посредством веб-браузера и технологии WebRTC. Практическая значимость исследования заключается в реализации собственного решения для связи в режиме реального времени с использованием технологии WebRTC. Новизна исследования состоит в том, что для реализации будут использованы open-source решения, что сделает реализованную системы связи абсолютно бесплатной в использовании.

Ключевые слова: видеопоток, передача видео, видеосвязь, WebRTC, Open Source.

METHODS OF IMPLEMENTING VIDEO COMMUNICATION IN A BROWSER USING OPEN-SOURCE TOOLS

Review article

Kuzminov I.V.[1, *]

[1] St. Petersburg State University of Industrial Technologies and Design, Saint-Petersburg, Russian Federation

* Corresponding author (ikuz.23[at]yandex.ru)

Abstract

Communication on the Internet has long ago reached a new level. Messengers and chats for exchanging short text messages with emoticons have been replaced by the possibility of making calls in audio and video format. And the merit here is primarily in the development of technology – it was the emergence of WebRTC technology that greatly simplified the implementation of this kind of communication for developers. The technology is free and easy to use, due to which it quickly became widespread. Within the framework of this article, a review of open-source solutions for real-time video communication through a web browser and WebRTC technology will be made. The practical significance of the research lies in the implementation of an in-house solution for real-time communication using WebRTC technology. The novelty of the study is that open-source solutions will be used for implementation, which will make the implemented communication system absolutely free to use.

Keywords: video streaming, video transmission, video communication, WebRTC, Open Source.

Введение

В рамках данной статьи представлено рассмотрение базового набора open-source инструментов, позволяющих реализовать инструментарий видеосвязи посредством обычного веб-браузера. В работе [2] представлен обзор возможностей протокола WebRTC, а также варианты реализации связи с его использованием. В исследовании [3] авторы предлагают разработку адаптивной платформы, основанной на использовании протокола WebRTC при организации взаимодействия участников образовательного процесса. В работе [6] представлено описание бесплатного сервера Kurento, позволяющего более обширно использовать возможности WebRTC. В работах [8] и [9] приводится описание процесса разработки систем видеоконференцсвязи на основании протокола WebRTC. Представленные работы демонстрируют высокую актуальность рассмотрения вопросов использования протокола WebRTC при организации видеосвязи посредством браузера. Тем не менее стоит отметить малое количество исследований, демонстрирующих практическое применение open source инструментов при работе с данным протоколом.

Реализация видеосвязи не является проблемой для разработчиков – достаточно взять один из распространенных браузеров, и сформировать для него веб-сервис с использованием стандарта WebRTC (англ. Web Real Time Communications – Веб коммуникации в режиме реального времени). Это открытый стандарт, который не требует особых вычислительных мощностей, и позволяет осуществлять обмен любой мультимедиа информацией между двумя и более пользователями, используя в качестве клиента веб-браузер. То есть пользователям не требуется установка дополнительных приложений или расширений для браузера – достаточно открыть страницу сервиса, разрешить доступ к камере и микрофону, и ожидать соединения либо осуществить вызов своего собеседника.

Технология WebRTC

Впервые WebRTC использовали в 2011 году, с целью реализации связи между двумя браузерам и передачи данных без каких-либо сторонних инструментов или сервисов. Обусловлено это было желанием компании Ericsson уйти от необходимости использования сторонних приложений или плагинов в составе веб-браузера при осуществлении связи. Ведь при наличии возможности мультимедийного взаимодействия между двумя браузерами упрощалась жизнь не только пользователей, но и разработчиков – не нужно было выполнять поддержку плагинов и следить за обновления браузеров, все это должно было работать на уровне функционала последнего [4]. За относительно небольшой промежуток времени WebRTC стала очень популярной технологией, и сегодня активно применяется в самых различных вариантах. В основном это, конечно же, передача потока видео и аудио на веб-платформах для видеосвязи, обучения, связи с удаленными камерами, либо это сервисы для передачи иной мультимедийной информации [8].

Функционирование данной технологии основано на использовании собственного набора API. Данные программные интерфейсы являются открытыми и бесплатными, но при этом довольно жестко стандартизированы. Это позволяет использовать их при реализации видеосвязи посредством любого современного браузера. Основой API технологии WebRTC является использование языка программирования JavaScript в совокупности с инструментами языка гипертекстовой разметки, что позволяет разработчикам использовать их для работы практически с любым современным веб-браузером. Для подтверждения данного факта стоит кратко описать, каков уровень поддержки технологии WebRTC в современных браузерах. Один из популярнейших сегодня браузеров Google Chrome содержит большой набор инструментов для отладки и тестирования работы сервисов видеосвязи на основе технологии веб-связи в режиме реального времени, что делает его наиболее популярным среди разработчиков. Браузер Mozilla Firefox также обладает очень хорошим инструментарием, обеспечивающим поддержку технологии WebRTC, а команда разработчиков Mozilla являются активными участниками сообщества, поддерживающего этот стандарт. Еще один из известных браузеров Opera тоже может похвастаться высоким уровнем поддержки технологии WebRTC, и наличием большого числа инструментов, позволяющих реализовывать собственные приложения с её использованием. Браузер Microsoft Edge, поставляемый в комплекте с операционной системой Windows 10 и 11-й версий, функционирует на движке Chromium, в связи с чем уровень поддержки им технологии WebRTC аналогичен браузеру Chrome. А аналогичный программный продукт от компании Apple – браузер Safari, поддержкой WebRTC обладает, однако при разработке решений для него рекомендуется проводить обязательную проверку совместимости [5]. Получается, что все популярные браузеры поддерживают технологию веб-связи в режиме реального времени, что подтверждает факт её высокой популярности сегодня.

То, для какого варианта организации связи с использованием технологии WebRTC будет реализовываться веб-сервис или приложение, оказывает непосредственное влияние на архитектуру данных программных решений, а также на используемые протоколы связи. Технология WebRTC может быть реализована с применением нескольких различных протоколов, различия в функциональном назначении между которыми довольно существенны: протокол MPEG-DASH/HLS используется при реализации сервисов потокового вещания по запросу, протокол WebRTC применяется при реализации обмена видео-потоком с низкой задержкой сигнала для связи между двумя или более браузерами, а протокол RTMP идентичен WebRTC с той разницей, что он заточен в большей степени на качественную передачу больших потоков информации, по причине чего он чаще применяется для реализации прямых трансляций.

Так как тема данной статьи напрямую связана с организацией видеосвязи в браузере, то протокол WebRTC представляет наибольший интерес для рассмотрения. Именно эта технология более всего подходит для видеосвязи в браузерах, и следует рассмотреть, с какими кодеками она работает. Так, для передачи аудио-потока используются кодеки Opus, G.711, G.722, PCMU и PCMA. Базовым кодеком передачи аудио является кодек Opus, при реализации приложений голосовой телефонии применяют кодеки G.711 и G.722, а для обеспечения поддержки устаревших приложений применяют кодеки PCMU и PCMA. Аналогичная ситуация и с кодеками для видеопотока. Базовым кодеком выступает VP8, обеспечивающий возможность динамического конфигурирования качества видео в зависимости от ширины канала связи. Помимо него могут быть использованы кодеки VP9, H.264 и H.265, но их использование варьируется в зависимости от используемого браузера или устройства, и по этой причине рекомендуется применять в работе только те кодеки, которые являются базовыми.

Функционирование технологии WebRTC

С учетом простоты реализации приложений и сервисов с применением WebRTC, сегодня их реализация чаще всего происходит на основании трех различных направлений:

- веб-соединение между браузерами двух пользователей, раньше подобное сетевое соединение часто называли «точка-точка»;

- веб-соединение, при котором медиапоток с одного браузера передается множеству браузеров, с которыми было установлено соединение. Яркий пример данного рода связи – трансляции видео, видеоуроки, стримы и т.п.

- веб-соединение с одновременным обменом медиа между несколькими браузерами. Чаще всего данные соединение используются в групповых звонках, видео-совещаниях и т.д.

Роль технологии WebRTC можно описать довольно просто – между двумя клиентами устанавливается соединение, при этом сервер используется только на этапе подключения пользователей (как это показано на рисунке 1). На всех этапах своей работы механизмы данной технологии решают очень много вопросов, связанных с разрывом соединения или потери данных, работы со скрытыми за NAT IP-адресами, настройки потока мультимедиа в соответствии с пропускной возможностью канала связи и т.п. Фактически, при реализации собственного приложения, разработчикам требуется реализовать процедуры вызова, начала и завершения соединения между абонентами, все остальное выполняют внутренние API [3].

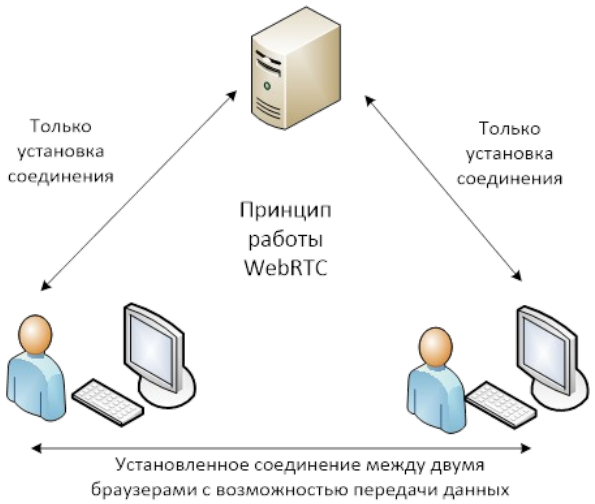

Рисунок 1 - Принцип функционирования технологии WebRTC
DOI: https://doi.org/10.23670/IRJ.2024.140.12.1

Представленный на рисунке 1 принцип работы технологии WebRTC описывают её лишь поверхностно, по причине чего необходимо выполнить более подробное рассмотрение её работы. Для этого следует рассмотреть пример осуществления связи между двумя браузерами. Пошагово данную процедуру можно описать в виде нескольких последовательных шагов:

- первый шаг – пользователь открывает страницу сервиса, который позволит ему организовать связь посредством WebRTC;
- второй шаг – в браузере появляется запрос на предоставление доступа к веб-камере и микрофону, на который пользователь должен ответить положительно;
- третий шаг – генерация пакета инициации соединения протоколом описания сеанса в браузере вызывающего абонента. Данный пакет представляет собой текстовый файл с описанием информации об устанавливаемом соединении;
- четвертый шаг – происходит передача сгенерированного пакета вызываемому абоненту, используя при этом сигнальный сервер и протокол WebSocket. При получении данного пакета в браузере принимаемого абонента выполняется генерация аналогичного пакета, и происходит его возврат вызываемому абоненту;
- пятый шаг – выполняется сетевое подключение между данными пользователями. Для этого они получают адрес сервера STUN, который будет работать над определением внешнего и внутреннего IP-адреса абонентов, а также дальнейшее формирование подключения между ними.
- шестой шаг – непосредственно установление сетевого соединения, при работе которого в рамках функционирования механизмов технологии выполняется вызов события «onicecandidate», позволяющего контролировать состояние текущего подключения [8].

Перечисленные шаги реализуются технологией WebRTC, а для обеспечения её работы разработчиками используются три API JavaScript:

- MediaStream (getUserMedia) – представляет собой интерфейс для передачи медиапотока;
- RTCPeerConnection – метод формирования связи между браузерами связывающихся абонентов. Он выполняет обработку всех сообщений с предложениями начала связи и ответы на эти сообщения, содержащие сформированные параметры начала сеанса;
- RTCDataChannel – метод, отвечающий за передачу любых типов данных в режиме онлайн. Данный метод идентичен с методами обмена данными на основании технологии WebSockets, однако фактически он выполняет подключение браузеров с целью организации прямого обмена информацией.

Описанные программные интерфейсы представляют собой основу технологии WebRTC, и по этой причине они поддерживаются всеми браузерами, представленными в данной статье. Однако для работы данной технологии существует уже большое число готовых бесплатных решений, которые позволяют существенно упростить процесс реализации видеосвязи в браузере между двумя абонентами. Рассмотрим наиболее известные Open-Source решения.

Одним из наиболее распространенных решений для реализации видеосвязи посредством браузера сегодня является платформа Zoom. Она представлена и приложением, и набором бесплатных программных интерфейсов, которые могут быть использованы для реализации собственных программных решений. Однако данный сервис был отмечен в нескольких исследованиях как обладающий низким уровнем защиты соединения, что ставит вопрос о целесообразности его использовании при реализации решений корпоративного уровня. Далее по уровню популярности следует сервис Ant Media Server, который по аналогии с платформой Zoom предоставляет инструментарий для реализации связи в аудио и видео формате в режиме реального времени. Это решение поддерживает большое число протоколов, предоставляет гибкое масштабирование реализуемых решений и может быть использовано в облачном варианте, либо установлено на локальный сервер. Платформа Kurento – это еще один сервис для реализации связи посредством браузера. Особенность этой платформы – высокий уровень контроля процедуры направления мультимедиапотока, что позволяет в дальнейшем более детально с ним работать, например в

случае организации работы технологии распознавания лиц и подобных её решений. Следующий пример Open-Source решения для реализации WebRTC – это платформа Open-EasyRTC. Она обладает всеми возможностями WebRTC, имеет большую библиотеку примеров реализации код для HTML5 и JavaScript, и довольно большой набор серверных и клиентских API. Еще одна бесплатная платформа – openVidu – распространяется по лицензии Apache License v2, имеет гибкую поддержку шифрования передаваемого мультимедиа потока, и может быть легко интегрирована в различные платформы JavaScript. Разработчикам предоставляется удобный набор программных интерфейсов REST. Протокол организации процедур связи между браузерами на основании сквозного шифрования SaltyRTC преимущественно используется при реализации зашифрованных приложений. Удобством для начала его использования на практике можно назвать большое число документации и примеров для разработчиков.

Помимо готовых сервисов и протоколов реализация прямой видеосвязи может быть выполнена с использованием специализированных веб-фреймворков или библиотек. Одним из распространенных решений является фреймворк Express, который может похвастаться незначительным объемом, но при этом существенным функционалом. При его применении разработчик получает возможность реализации как веб-приложений, так и мобильных приложений для реализации прямой видеосвязи на основе WebRTC. Он имеет собственные API, посредством которых разработчики могут работать как со служебными методами HTTP, так и с функционалом Node.js. Еще одним примером является проект Temasys, основанный на библиотеке SkylinkJS. Он имеет большой объем документации, согласно которой разработчики могут работать с различными технологиями – Swift, Objective-C, React, C++, Java и Android. Следующий пример – JSCommunicator – решение на основе языка JavaScript, которое использует HTML и CSS. Функционирует оно на основании протокола SIP и технологии WebSockets, и обладает высоким уровнем совместимости как с различными веб-приложениями, так и платформами систем управления контентом. Последним примером инструментария для работы с WebRTC будет представлена библиотека PeerJS, представляющая собой довольно простой API. Работает данная библиотека на основе JavaScript, обладает совместимостью с другими фреймворками, основанными на языке программирования JavaScript [1].

После того как были рассмотрены готовые бесплатные решения для WebRTC, следует описать процесс реализации собственного приложения для видеосвязи в режиме реального времени с использованием open-source инструментов. Для этого в первую очередь следует произвести установку данного решения. Так как речь идет о бесплатных решениях, то в качестве серверной платформы будем рассматривать Linux, например, Ubuntu Server.

Пример программного кода для реализации сервера сигналинга WebRTC

Установка любого сервера происходит в несколько этапов. Сначала необходимо установить среду выполнения приложения сервера. В данном примере это будет среда выполнения JavaScript – Node.js. Web-сервер сигналинга будет реализован с использованием фреймворка Express. Пример программного кода для реализации сервера сигналинга WebRTC:

```
        // Запуск TURN сервера (в качестве TURN сервера можно использовать open-source решение node-
turn)
        turnServer.start();
        // Создаем объект приложения Express
        export const app = express();
        // Привязываем обработчик на путь /websocket
        app.use('/websocket', (req, res) => {
            // Добавляем заголовки для переключения соединения с HTTP на WebSocket
            res.append('Upgrade', 'websocket');
            res.append('Connection', 'upgrade');
            // Добавляем заголовок с шифрованным ключом для безопасности соединения
            const shasum = crypto.createHash('sha1');
            shasum.update(req.headers['sec-websocket-key'] + 'key');
            const secHeader = shasum.digest('base64');
            res.append('Sec-WebSocket-Accept', secHeader);
            // Возвращаем код ответа протокола HTTP - 101, который сообщает о переключении протокола
            return res.status(101);
        });
        // Создаем сервер WebSocket
        const webSocketServer = new WebSocketServer({port: 9124});
        // Текущие соединения
        const connections = {};
        let currentId = 0;
        // Добавляем обработчик событий connection
        webSocketServer.on('connection', connection => {
            // Сохраняем соединения в объект соединений
            connections[currentId] = connection;
            connection.cliendId = currentId;
            currentId++;
            // Для каждого существующего соединения отправляем данные о новом клиенте
            Object.keys(connections).forEach(id  =>  connections[id].send(JSON.stringify({type:  'id',
id: connection.cliendId})));
            // Обработчик сообщений соединения
            connection.on('message', function(message) {
                // Парсинг сообщения
                const msg = JSON.parse(message);
                // Отправка сообщения соответствующему пользователю
                connections[msg.target]?.send(message);
            });
            // Обработчик закрытия соединения
            connection.on('close', () => {
                // Удалить соединение из списка соединений
                delete connections[connection.cliendId];
            });
        });
        const PORT = 9123;
        // Слушаем указанный порт
        app.listen(PORT, '127.0.0.1');
```

Рисунок 2 - Пример программного кода для реализации сервера сигналинга WebRTC

DOI: https://doi.org/10.23670/IRJ.2024.140.12.2

Рассмотрим процесс реализации клиента WebRTC. Сначала устанавливается соединение с сервером сигналинга через WebSocket. После чего добавляются обработчики сообщений с сервера для корректной установки соединения между абонентами. Ниже приведен пример реализации клиентского кода для осуществления видеозвонка в браузере.

```
        // Открываем WebSocket соединение
        const wsConnection = new WebSocket('ws://localhost:9124');
        // Создаем необходимые переменные
        let clientId;
        let targetId = null;
        let myPeerConnection = null;
        let transceiver = null;
        let webcamStream = null;
        // Ограничения медиа потока
        const mediaConstraints = {audio: true, video: {aspectRatio: {ideal: 1.333333}}};
        function sendToServer(msg) {
            const msgJSON = JSON.stringify(msg);
            wsConnection.send(msgJSON);
        }
        function closeVideoCall() {
            // Получаем объект локального видео
            const localVideo = document.getElementById('local_video');
            // Сбрасываем данные peer соединения
            myPeerConnection.onnicecandidate = null;
            myPeerConnection.oniceconnectionstatechange = null;
            myPeerConnection.onsignalingstatechange = null;
            myPeerConnection.onnotificationneeded = null;
            // Останавливаем все приемопередатчики
            myPeerConnection.getTransceivers().forEach(transceiver => transceiver.stop());
            if (localVideo.srcObject) {
                // Останавливаем видео поток
                localVideo.pause();
                localVideo.srcObject.getTracks().forEach(track => track.stop());
            }
            // Закрываем соединение
            myPeerConnection.close();
            // Сбрасываем значения переменных
            myPeerConnection = null;
            webcamStream = null;
            targetId = null;
        }
        function hangUpCall() {
            // Завершить видео звонок
            closeVideoCall();
            // Отправить серверу сигналинга сообещение о завершении звонка
            sendToServer({ target: targetId, type: 'hang-up'});
        }
        async function handleNegotiationNeededEvent() {
            // Создаем предложение для соединения
            const offer = await myPeerConnection.createOffer();
            // Если состояние сигналинга не стабильное завершим обработку
            if (myPeerConnection.signalingState !== 'stable') return;
            // Создаем локальное описание для предложения
```

Рисунок 3 - Процесс реализации клиента WebRTC 1
DOI: https://doi.org/10.23670/IRJ.2024.140.12.3

```
            await myPeerConnection.setLocalDescription(offer);
            // Отправка предложения удаленному участнику
            sendToServer({clientId, target: targetId, type: 'video-offer', sdp:
myPeerConnection.localDescription});
        }
        function handleICEConnectionStateChangeEvent() {
            switch (myPeerConnection.iceConnectionState) {
                case 'closed':
                case 'failed':
                case 'disconnected':
                    closeVideoCall();
                    break;
            }
        }
        // Обработчик изменения состояния сигналинга
        function handleSignalingStateChangeEvent() {
            switch (myPeerConnection.signalingState) {
                case 'have-remote-offer': {
                    handleVideoOfferMsg(myPeerConnection.msg);
                    break;
                }
                case 'closed':
                    closeVideoCall();
                    break;
            }
        }
        function createPeerConnection() {
            // Установка соединения
            myPeerConnection = new RTCPeerConnection({iceServers: [{urls: 'turn:localhost:7788',
username: 'webrtc', credential: 'turnserver'}]});
            // Устанавливаем обработчики peer соединения
            myPeerConnection.onicecandidate = e => sendToServer({type: 'new-ice-candidate',
target: targetId, candidate: e.candidate});;
            myPeerConnection.oniceconnectionstatechange = handleICEConnectionStateChangeEvent;
            myPeerConnection.onsignalingstatechange = handleSignalingStateChangeEvent;
            myPeerConnection.onnegotiationneeded = handleNegotiationNeededEvent;
        }
```

Рисунок 4 - Процесс реализации клиента WebRTC 2
DOI: https://doi.org/10.23670/IRJ.2024.140.12.4

```
        // Обработчик сообщений WebSocket
        wsConnection.onmessage = async(event) => {
            // Получение данных
            const data = await new Response(event.data).json();
            switch (data.type) {
                case 'video-offer': {
                    // Обработать предложение обмена видео
                    handleVideoOfferMsg(data);
                    break;
                }
                case 'video-answer': {
                    // Обработать видео ответ
                    handleVideoAnswerMsg(data);
                    break;
                }
                case 'new-ice-candidate': {
                    // Обработать сообщение о новом ICE кандидате
                    handleNewICECandidateMsg(data);
                    break;
                }
                case 'hang-up': {
                    // Завершить звонок
                    closeVideoCall();
                    break;
                }
            }
        }
        // Функция приглашения
        async function invite(evt) {
            // Получаем id целевого участника
            targetId = evt.target.textContent;
            // Установка соединения для приглашения участника
            createPeerConnection();
            // Получаем доступ к камере и микрофону текущего пользователя и перенаправляем поток
в видео объект на странице
            webcamStream = await navigator.mediaDevices.getUserMedia(mediaConstraints);
            document.getElementById('local_video').srcObject = webcamStream;
            // Добавляем приемопередатчики
            webcamStream.getTracks().forEach(transceiver = track =>
myPeerConnection.addTransceiver(track, {streams: [webcamStream]}));
        }
```

Рисунок 5 - Процесс реализации клиента WebRTC 3
DOI: https://doi.org/10.23670/IRJ.2024.140.12.5

```
        // Функция-обработчик видео предложения
        async function handleVideoOfferMsg(msg) {
            targetId = msg.clientId;
            // Создание описания (SDP) сессии
            const desc = new RTCSessionDescription(msg.sdp);
            // Устанавливаем SDP
            await myPeerConnection.setRemoteDescription(desc);
            // Создаем и отправляем звонящему ответ на звонок
            Await myPeerConnection.setLocalDescription(await
myPeerConnection.createAnswer()).catch(e => console.error(e));
            sendToServer({target: targetId, type: 'video-answer', sdp:
myPeerConnection.localDescription});
        }
        async function handleVideoAnswerMsg(msg) {
            // Получатель вызова принял наш звонок, обрабатываем данные
            const desc = new RTCSessionDescription(msg.sdp);
            await myPeerConnection.setRemoteDescription(desc);
        }
        async function handleNewICECandidateMsg(msg) {
            // Добавляем полученного ICE кандидата
            const candidate = new RTCIceCandidate(msg.candidate);
            await myPeerConnection.addIceCandidate(candidate)
        }
```

Рисунок 6 - Процесс реализации клиента WebRTC 4
DOI: https://doi.org/10.23670/IRJ.2024.140.12.6

Для получения доступа к камере и микрофону пользователя в данном коде используется метод getUserMedia, посредством которого происходит реализация объекта RTCPeerConnection. После его создания будет добавлен локальный поток медиа-информации с обязательным соединением и созданием оффера для начала соединения. Затем начинается воспроизведение собственного потока на открытой странице и происходит переход в режим ожидания соединения с другими абонентами. Для того чтобы обнаружить новых кандидатов ICE и обработать потоки медиа от удаленных клиентов, в коде размещены обработчики соответствующих событий. После того как соединение будет установлено, на странице дополнительно будет воспроизведён поток мультимедиа от собеседника.

Последним примером будет приведена реализация простейшего клиента WebRTC с использованием фреймворка EasyRTC:

```
// Создаем объект easyrtc и подключаемся к серверу
easyrtc.setSocketUrl(«https://localhost:9124»);
easyrtc. connect («myApp», () => {
  console.log («Connected to EasyRTC server»);});
// Получаем доступ к видеокамере и микрофону
easyrtc.enableAudio(true);
easyrtc.enableVideo(true);
// Обработчик события получения вызова
easyrtc.setStreamAcceptor((callerEasyrtcid, stream) => {
  // Получаем доступ к видео-элементу на странице
  const video = document.querySelector('video');
  // Воспроизводим видеопоток собеседника на странице
  easyrtc.setVideoObjectSrc(video, stream);
});
// Обработчик события нажатия кнопки «Вызвать»
document.querySelector('#call-btn').addEventListener('click', () => {
  // Получаем ID собеседника из текстового поля
  const peerId = document.querySelector('#peer-id').value;
  // Вызываем собеседника
  easyrtc.call(peerId, () => {
    console.log ("Call accepted by " + peerId);
  }, (errorMessage) => {
    console.error("Error calling " + peerId + ": " + errorMessage);  });});});
```

Рисунок 7 - Реализация простейшего клиента WebRTC с использованием фреймворка EasyRTC
DOI: https://doi.org/10.23670/IRJ.2024.140.12.7

Представленный пример фактически отличается от предыдущего незначительно – точно также устанавливается соединение с сервером, происходит получение ID клиента, далее предоставляется доступ к устройствам захвата мультимедиа, и происходит ожидание соединения с другим клиентом, либо вызов другого абонента.

Заключение

Резюмируя, можно сделать вывод, что сегодня на рынке программного обеспечения существует большое количество open-source решений, которые могут быть использованы для реализации собственного приложения WebRTC. При этом их функционал и возможности ограничены лишь опытом специалиста, который будет работать с ними. Рассмотренные примеры продемонстрировали, что базовый набор функций при реализации клиента практически одинаков, а принципы их функционирования абсолютно схожи. Но тем не менее, при выборе конечного решения следует внимательно изучить документацию и возможности выбираемого фреймворка, а также оценивать, насколько он может подойти для реализуемого сервиса или приложения.

Конфликт интересов

Не указан.

Рецензия

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

Conflict of Interest

None declared.

Review

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

Список литературы / References

1. Quality of Experience Estimation for WebRTC-based Video Streaming / N. Amram, Y. Sulema, O. Aleshchenko [et al.] // 24th European Wireless 2018 "Wireless Futures in the Era of Network Programmability", EW 2018: 24, Wireless Futures in the Era of Network Programmability, Catania, 02–04 May 2018. — Catania, 2018. — P. 94-99.

2. Абдурайимов Л.Н. Webrtc как набор API, предназначенных для организации передачи потоковых данных между браузерами / Л.Н. Абдурайимов, Д.Р. Могильный // Современные научные исследования: Сборник научных трудов по материалам IX Международной научно-практической конференции, Анапа, 17 июня 2019 года. — Анапа: Общество с ограниченной ответственностью «Научно-исследовательский центр экономических и социальных процессов» в Южном Федеральном округе, 2019. — С. 71-76.

3. Ушаков Ю.А. Адаптивная платформа видеоконференцсвязи на основе WebRTC в интернет образовании / Ю.А. Ушаков, П.Н. Полежаев, А.Е. Шухман [и др.] // Современные информационные технологии и ИТ-образование. — 2019. — Т. 15, № 3. — С. 746-754.

4. Борг Н. Как технологии WebRTC и SIP дополняют друг друга в сервисе PBX Express / Н. Борг // Системный администратор. — 2017. — № 11(180). — С. 54-55.

5. Гуля И.А. Обзор возможностей протокола WebRTC для организации связи в реальном времени / И. А. Гуля // Перспективные информационные и телекоммуникационные технологии: Российская научно-техническая конференция аспирантов и молодых ученых, посвященная 20-летию СО МАИ, Новосибирск, 25–26 февраля 2016 года. — Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2016. — С. 38-41.

6. Ловаков Р.И. Исследование возможностей WebRTC медиа сервера Kurento / Р.И. Ловаков, Ю.А. Холодий // Научные открытия 2018: Материалы XXXVIII Международной научно-практической конференции, Москва, 28–25 мая 2018 года. — Москва: Научный центр "Олимп", 2018. — С. 64-74.

7. Мороз И. Д. Потоковое вещание медиафайлов в одноранговой сети с помощью webrtc / И. Д. Мороз // Молодежный научный форум: технические и математические науки. — 2016. — № 7(36). — С. 11-15.

8. Ушаков Ю. А. Повышение качества восприятия информации при проведении образовательных многоточечных видеоконференций на основе технологии webrtc / Ю. А. Ушаков, А. Л. Коннов, Н. Ю. Ушакова // Научно-технический вестник Поволжья. — 2019. — № 12. — С. 163-167.

9. Фоминский А.С. Разработка системы видеоконференций с использованием технологии WebRTC / А.С. Фоминский // МНСК-2018: Информационные технологии: Материалы 56-й Международной научной студенческой конференции, Новосибирск, 22–27 апреля 2018 года. — Новосибирск: Новосибирский национальный исследовательский государственный университет, 2018. — С. 203.

10. Ширяев А.А. Технологии webrtc и ORTC как замена видеотрансляций на Flash / А.А. Ширяев, М. В. Глущенко // Общество – наука – инновации: сборник статей по итогам Всероссийской научно-практической конференции, Иркутск, 22 декабря 2019 года. Том Часть 1. — Иркутск: Общество с ограниченной ответственностью "Агентство международных исследований", 2019. — С. 99-103.

Список литературы на английском языке / References in English

1. Quality of Experience Estimation for WebRTC-based Video Streaming / N. Amram, Y. Sulema, O. Aleshchenko [et al.] // 24th European Wireless 2018 "Wireless Futures in the Era of Network Programmability", EW 2018: 24, Wireless Futures in the Era of Network Programmability, Catania, 02–04 May 2018. — Catania, 2018. — P. 94-99.

2. Abduraimov L.N. Webrtc kak nabor API, prednaznachennyh dlya organizacii peredachi potokovyh dannyh mezhdu brauzerami [Webrtc as a Set of APIs Intended for Organizing the Transfer of Streaming Data between Browsers] / L. N. Abduraimov, D. R. Mogilny // Sovremennye nauchnye issledovaniya: Sbornik nauchnyh trudov po materialam IX Mezhdunarodnoj nauchno-prakticheskoj konferencii, Anapa, 17 iyunya 2019 goda [Modern Scientific Research: Collection of scientific papers based on the materials of the IX International Scientific and Practical Conference, Anapa, June 17, 2019]. — Anapa: Limited Liability Company “Research Center for Economic and Social Processes” in the Southern Federal District, 2019. — p. 71-76. [in Russian]

3. Ushakov YU.A. Adaptivnaya platforma videokonferencsvyazi na osnove WebRTC v internet obrazovanii [Adaptive Video Conferencing Platform Based on WebRTC in Internet Education] / Yu.A. Ushakov, P.N. Polezhaev, A.E. Shukhman [et al.] // Sovremennye informacionnye tekhnologii i IT-obrazovanie [Modern Information Technologies and IT Education]. — 2019. — Vol. 15, No. 3. — P. 746-754. [in Russian]

4. Borg N. Kak tekhnologii WebRTC i SIP dopolnyayut drug druga v servise PBX Express [How WebRTC and SIP Technologies Complement Each Other in the PBX Express Service] / N. Borg // Sistemnyj administrator [System Administrator]. — 2017. — No. 11(180). — p. 54-55. [in Russian]

5. Gulya I.A. Obzor vozmozhnostej protokola WebRTC dlya organizacii svyazi v real'nom vremeni [Review of the Capabilities of the WebRTC Protocol for Organizing Communication in Real Time] / I. A. Gulya // Perspektivnye informacionnye i telekommunikacionnye tekhnologii: Rossijskaya nauchno-tekhnicheskaya konferenciya aspirantov i molodyh uchenyh, posvyashchennaya 20-letiyu SO MAI, Novosibirsk, 25–26 fevralya 2016 goda [Advanced Information and Telecommunication Technologies: Russian Scientific and Technical Conference of Graduate Students and Young Scientists, dedicated to the 20th anniversary of SO MAI, Novosibirsk, February 25–26, 2016]. — Novosibirsk: Siberian State University of Telecommunications and Informatics, 2016. — P. 38-41. [in Russian]

6. Lovakov R.I. Issledovanie vozmozhnostej WebRTC media servera Kurento [Study of the Capabilities of WebRTC Media Server Kurento] / R.I. Lovakov, Yu.A. Kholodiy // Nauchnye otkrytiya 2018: Materialy XXXVIII Mezhdunarodnoj nauchno-prakticheskoj konferencii, Moskva, 28–25 maya 2018 goda [Scientific Discoveries 2018: Proceedings of the XXXVIII International Scientific and Practical Conference, Moscow, May 28–25, 2018]. — Moscow: Scientific Center “Olympus”, 2018. — P. 64-74. [in Russian]

7. Moroz I. D. Potokovoe veshchanie mediafajlov v odnorangovoj seti s pomoshch'yu webrtc [Streaming Media Files in a Peer-to-peer Network Using Webrtc] / I. D. Moroz // Molodezhnyj nauchnyj forum: tekhnicheskie i matematicheskie nauki [Youth Scientific Forum: Technical and Mathematical Sciences]. — 2016. — No. 7(36). — p. 11-15. [in Russian]

8. Ushakov Yu. A. Povyshenie kachestva vospriyatiya informacii pri provedenii obrazovatel'nyh mnogotochechnyh videokonferencij na osnove tekhnologii webrtc [Improving the Quality of Information Perception When Conducting Educational Multipoint Video Conferences Based on Webrtc Technology] / Yu. A. Ushakov, A. L. Konnov, N. Yu. Ushakova // Nauchno-tekhnicheskij vestnik Povolzh'ya [Scientific and Technical Bulletin of the Volga Region]. — 2019. — No. 12. — P. 163-167. [in Russian]

9. Fominsky A.S. Razrabotka sistemy videokonferencij s ispol'zovaniem tekhnologii WebRTC [Development of a Video Conferencing System Using WebRTC Technology] / A.S. Fominsky // MNSK-2018: Informacionnye tekhnologii: Materialy 56-j Mezhdunarodnoj nauchnoj studencheskoj konferencii, Novosibirsk, 22–27 aprelya 2018 goda [MNSK-2018: Information Technologies: Proceedings of the 56th International Scientific Student Conference, Novosibirsk, April 22–27, 2018]. — Novosibirsk: Novosibirsk National Research State University, 2018. — P. 203. [in Russian]

10. Shiryaev A.A. Tekhnologii webrtc i ORTC kak zamena videotranslyacij na Flash [WebRTC and ORTC Technologies as a Replacement for Video Broadcasts on Flash] / A.A. Shiryaev, M. V. Glushchenko // Obshchestvo – nauka – innovacii: sbornik statej po itogam Vserossijskoj nauchno-prakticheskoj konferencii, Irkutsk, 22 dekabrya 2019 goda [Society – Science – Innovation: collection of articles based on the results of the All-Russian Scientific and Practical Conference, Irkutsk, December 22, 2019]. Volume Part 1. — Irkutsk: Limited Liability Company "Agency for International Research", 2019. — P. 99-103. [in Russian]