DOI: 10.25198/1814-6457-221-109
УДК 378.018.43

Ушаков Ю.А., Шухман А.Е., Парфенов Д.И., Полежаев П.Н., Ушакова М.В.
Оренбургский государственный университет, г. Оренбург, Россия
E-mail: shukhman@gmail.com

МЕТОДИЧЕСКИЕ АСПЕКТЫ ИСПОЛЬЗОВАНИЯ ПЛАТФОРМЫ ПРОВЕДЕНИЯ ВИДЕОКОНФЕРЕНЦИЙ В СИСТЕМЕ НЕПРЕРЫВНОГО ОБРАЗОВАНИЯ

Технологию WebRTC стали поддерживать все основные браузеры, многие системы видеоконференцсвязи перешли на WebRTC как на основную технологию. Но до сих пор остаются проблемы контроля качества голоса и видео, особенно в многоточечных конференциях, где нет времени контролировать каждого участника и нет возможности быстро переподключить часть участников. Работа посвящена возможности автоматического объективного контроля качества восприятия на клиентской стороне как для аудио, так и для видео потока для оперативного решения проблем передачи WebRTC.

Ключевые слова: WebRTC, профессиональное образование, многоточечная видеосвязь, индивидуальное обучение.

Для цитирования: Методические аспекты использования платформы проведения видеоконференций в системе непрерывного образования / Ю.А. Ушаков, А.Е. Шухман, Д.И. Парфенов, П.Н. Полежаев, М.В. Ушакова // Вестник Оренбургского государственного университета. – 2019. – №3(221). –С. 109–117. DOI: 10.25198/1814-6457-221-109.

Ushakov YU.A., SHukhman A.E., Parfenov D.I., Polezhaev P.N., Ushakova M.V.
Orenburg state university, Orenburg, Russia
E-mail: shukhman@gmail.com

METHODOLOGICAL ASPECTS OF USING THE VIDEOCONFERENCING PLATFORM IN THE CONTINUING EDUCATION SYSTEM

All major browsers began to support WebRTC technology, many video conferencing systems switched to WebRTC as the main technology. But there still remain problems of controlling the quality of voice and video, especially in multi-point conferences, where there is no time to control each participant and there is no way to quickly reconnect some of the participants. The work is devoted to the possibility of automatic objective quality control of perception on the client side for both audio and video streams for the operational solution of WebRTC transmission problems.

Key words: WebRTC, professional education, multi-point video communication, individual training.

For citation: Ushakov YU.A., SHukhman A.E., Parfenov D.I., Polezhaev P.N., Ushakova M.V. Methodological aspects of using the videoconferencing platform in the continuing education systemю *Vestnik Orenburgskogo gosudarstvennogo universiteta*, 2019, no. 3(221), pp. 109–117. DOI: 10.25198/1814-6457-221-109.

Активное внедрение информационных технологий (ИТ) во все сферы человеческой деятельности предопределяет необходимость использования современных форм подготовки обучающихся. Сегодня актуальным является новая тенденция образования, основанная на двух направлениях:

1) развитие открытого, дистанционного образования, технологической основой которого являются информационные и телекоммуникационные технологии;

2) универсализация содержания и методик обучения, что решается посредством широкого внедрения и развития электронных форм передачи материала.

Дистанционное обучение позволяет учиться в своем собственном темпе, исходя из своих потребностей в образовании и личностных особенностей. Так же оно позволяет не ограничивать себя в выборе образовательного учреждения, независимо от того, в каком регионе проживает обучающийся. В процессе дистанционного обучения используются современные технологии, что также позволяет освоить навыки, которые в будущем пригодятся в работе и повседневной жизни. Одним из самых главных удобств является возможность самим корректировать и составлять график обучения, расписание занятий, а также список изучаемых предметов.

Нельзя не отметить еще одно достоинство – это обучение в максимально комфортной и привычной обстановке, что способствует повышению его продуктивности.

Мониторинг форм, методов и средств дистанционного обучения в Оренбургской области, проведенный в сентябре-октябре 2019 г. показал, что для образовательных учреждений существует высокая потребность в таких ресурсах. Так, например в 35% школ активно используют в работе видео данные, в 18% школ ежедневно обращаются к информационным ресурсам для интерактивного выполнения заданий, а 5% имеют собственные площадки, формирующие электронную образовательную среду, но лишь 2% образовательных учреждений используют системы ВКС для непосредственной организации образовательной деятельности.

Опрос исследования показал, что наиболее востребованным ресурсом является образовательная платформа. Около 87% опрошенных указали, что им было бы удобно размещать образовательные материалы и проводить занятия с поддержкой видеоконтента. Системой вебинаров готовы пользоваться 45% образовательных учреждений. Размещать ресурсы для обеспечения образовательной деятельности на собственной платформе готовы 54% учителей, из них 23% уже имеют положительный опыт использования подобных платформ в практике образовательного процесса.

К сожалению используемые платформы для видеоконференций не поддерживают необходимый функционал: проведение видеолекций и вебинаров.

Видеолекции являются техническим средством активизации, организации и управления познавательной деятельностью студентов. Использование видеолекций позволяют повысить эффективность учебного процесса за счет:

– виртуального присутствия на предприятиях и в учреждениях по профилю будущей специальности, в научных лабораториях, экспедициях и т. п.;

– показа уникальных или быстропротекающих явлений, процессов, событий, «эффекта присутствия» при демонстрации «живых», реальных явлений или их виртуальных моделей;

– зрительного соучастия в предъявляемых реальных (или специально поставленных) ситуациях выбора решения управленческой или производственной проблемы;

– перемены видов деятельности, переключения внимания и использования как рационально-логического, так и эмоционально-образного мышления.

Как правило, видеолекции заранее записываются лектором и используются студентами в рамках индивидуальной самостоятельной работы. Достоинствами таких лекций является возможность повторить или наверстать пропущенный материал, самостоятельно более глубоко изучить новую тему, доступность и экономия времени. Недостаток – отсутствие непосредственной обратной связи с преподавателем.

В зависимости от технологии записи можно выделить следующие виды видеолекций:

1 Видеозапись лектора («говорящая голова»).

2 Живая запись лекции (запись обычной лекции в аудитории, где проходит занятие со студентами).

3 Видеолекция-интервью (запись интервью с лектором) .

4 Студийная видеолекция (запись постановочной лекции в студии).

5 Слайд-лекция, видеолекция, записанная на видеокамеру, видеолекция, записанная с экрана монитора средствами специальных программ захвата видеоизображения (такие лекции характеризуются полным отсутствием визуально-психологического контакта с лектором, но обязательно сопровождаются закадровым голосом диктора) .

6 Интерактивные видеолекции с синхронными слайдами (монолог лектора сопровождается слайдами с материалами, присутствуют кнопки управления показом, автоматизированный контроль знаний).

Расширением видеолекций является система вебинаров. Вебинары позволяют общаться с другими учениками и взаимодействовать с ними и преподавателем в процессе обучения, в котором можно обмениваться информацией и документами в режиме реального времени.

Этот тип ресурса очень полезен, поскольку, как и бесплатные онлайн-курсы, он предоставляет доступ к очень полезному контенту и информации о дисциплине, которую мы хотим изучать, без необходимости платить евро,

а также позволяет узнать о новых услугах и компаниях, специализирующихся на определенных Секторы. Вебинар является семинаром, транслируемым в режиме реального времени в сети Интернет. Его можно использовать как для обучения, наставничества, так и просто для общения между спикерами с участием посетителей, которые в некоторых случаях могут задавать вопросы и взаимодействовать с спикерами. Они могут быть записаны заранее или в прямом эфире, это семинары, конференции и переговоры, которые передаются через Интернет через платформы, которые позволяют это взаимодействие.

Вебинар – это мощный инструмент общения в виртуальной среде, широко используемый в образовании. Тем не менее, потенциал этого инструмента для продвижения значимого обучения часто недооценивается. Часто вебинар используется в качестве простой однонаправленной веб-трансляции. Тем не менее, в настоящее время существует тенденция считать его отличным ресурсом для построения обучения, диалогов и встреч.

Создаются совместные учебные среды, которые служат для конструктивного взаимодействия и обучения между сверстниками, в то же время давая свободу выбора индивидуальных туров самостоятельно. Вебинар может способствовать виртуальному образованию до тех пор, пока принимаются педагогические решения (кто участвует, как они связаны, кто сопровождает их в туре, какое содержание разрабатывается) и сформулированы в сборке онлайн-платформы, которая позволяет определить виды деятельности предложил.

Проведение занятия в режиме вебинара имеет следующие достоинства:

– доступность технологии (для проведения вебинара участнику необходим только доступ к сети Интернет и любой интернет-браузер);

– интерактивность (все действия преподавателя отображаются в режиме реального времени на компьютерах студентов);

– обратная связь со студентами (система опроса, а также встроенный в приложение чат, который позволяет преподавателю и студентам обмениваться сообщениями);

– запись проводимых семинаров (занятие в режиме вебинара может быть записано и выложено в виде отдельного электронного ресурса, доступного для скачивания и просмотра в офлайн-режиме).

К сожалению используемые платформы для видеоконференций не поддерживают необходимый функционал: проведение видеолекций и вебинаров с автоматическим контролем качества связи.

Методы диагностики проблем видеосвязи

Передача голоса и видео по сетям передачи имеет долгую историю и современные распространенные проблемы со всеми широко используемыми способами передачи происходят из старых телефонных подходов к передаче чувствительного к задержкам трафика.

Передачу голоса в IP сетях сейчас называют Voice over IP (VoIP) и в основном осуществляют на основе протокола Real-time Transport Protocol (RTP) с различными надстройками. Но телефония это не только передача голоса, но и сигнализация о вызовах, согласование и синхронизация кодеков, служебной информации, таймеров. Для телефонии поверх IP сетей (IP телефонии) используются протоколы сигнализации, которые также, как и протоколы ISDN используют выделенный канал сигнализации (SIP, H.323, IAX).

WebRTC использует протокол SRTP (Secure Real-time Transport Protocol) для передачи данных (не только голоса и видео, но и любых других), а в качестве сигнализации и синхронизации сторон необходимо использовать набор почти независимых друг от друга технологий, внешних и внутренних компонентов. Сигнализация не специфицирована и зависит от сервера и протокола, который ее обеспечивает, потоки RTP должны быть согласованы между точками приема и передачи как по кодекам, так и по типу содержимого пакетов RTP (RTP payload), по портам UDP и TCP (при прямой передаче), по использованию средств обнаружения (STUN) и обхода NAT (TURN). Если один из компонентов не отработал корректно, закрыты порты брандмауэром, отсутствуют промежуточные сервера STUN/TURN, может не быть голоса и одной из сторон, пропасть видео, внезапно прерваться вся связь.

Диагностика проблем связи установления соединения

WebRTC очень децентрализованный, поэтому основная диагностика должна быть доступна на клиентской стороне. Однако сетевые проблемы, связанные с NAT и брандмауэром нужно решать только совместно с инструментами на обеих сторонах. Проект [2] позволяет протестировать работу WebRTC на каждом шаге, но он не позволяет проверить открытость конкретных портов и технологий, а также не позволяет потом использовать результаты в конкретных приложениях. Тестирование сети состоит из следующих этапов

Тестирование доступа из браузера к порту TCP сервера сигнализации по протоколу HTTPS/Websocket – осуществляется простым запросом Ajax из javascript, при этом «xhr.timeout» должен устанавливаться в минимальное значение (например 300 мс) и только при неуспешном тесте прогрессивно увеличиваться. Это позволит одновременно узнать задержку в канале до сервера сигнализации и использовать ее потом в приложении. WebSocket и Long Poll запросы требуют использования setTimeout для прерывания соединения дольше заданного, для этого можно использовать javascript библиотеку okHttp или подобную. В работе использовалась okHttp3.

Тестирование доступа из браузера к порту TCP и UDP требуемых серверов STUN/TURN, тестирование авторизации на них. Для тестирования используются компоненты «window.RTCPeerConnection» и аналоги для браузеров Firefox и Safari [3]. После успешного запроса сервер TURN должен выдать порт и адрес прокси для RTP, который требуется также протестировать.

Тестирование доступа к порту UDP сервера коммутации WebRTC (SFU или MCU) и передачи/приема (эхо) RTP потока с требуемыми payload. Для этого используется второе соединение к TURN серверу, затем создается канал данных, по которому передается тестовое сообщение через метод «peer.send». Используется подход и библиотека simple-peer [4], таким образом проверяется P2P соединение через TURN между браузерами на стороне клиента.

Диагностика проблем во время сеанса

Диагностика в реальном времени самая трудная, потому что недостаточно узнать о потере пакета, требуется знать реальное воздействие на звук/изображение и степень ухудшения субъективного качества (QoE). Поэтому используется следующие подходы:

Проверка показателей WebRTC getStats (googJitterReceived, googCurrentDelayMs, packetsLost, bitsReceivedPerSecond, googDecodeMs, googEncodeUsagePercent, googRtt, googNacksSent, googNacksReceived)

Периодические синхронные скриншоты принимаемого и отправляемого изображения на клиентах, отсылка на сервер для сравнения с качества. Этот подход должен учитывать текущую задержку передачи чтобы сравнивать одинаковые фреймы, для минимизации влияния задержки можно захватывать несколько фреймов и отправлять их обратно на сервер в виде усредненных изображений.

Модификация изображения через использование своего метода создания потока [5] для добавления невидимых или маловидимых меток на изображение для контроля на удаленной стороне их наличия и соответствия. Для этого поток, полученный из видеокамеры через getUserMedia ассоциируется непосредственно со скрытым элементом «video», который модифицируется средствами рисования на канвасе, потом получившийся поток через захват Canvas.captureStream ассоциируется с нужным PeerConnection. Это может происходить только периодически (для реальных ситуаций), или постоянно (для тестов).

Качество звука сложно контролировать без вмешательства в передачу, для этого используются объекты AudioContext – AudioBufferSourceNode для генерации звуковой метки (шум, волна, щелчки…), AnalyserNode – для поиска в звуке этой метки. Способы генерации и анализа звука выходят за рамки этой статьи.

Решение проблем связи WebRTC

Основной проблемой с доступом к портам серверов является недоступность из за брандмауэра. Если на маршрутизаторе запрещены только некоторые из портов, можно воспользоваться обходными способами:

При разрешенных только TCP 443/80 портах (максимально закрытый вариант) необходимо принимать комплекс мер. На TURN сервере для таких клиентов нужно использовать только TCP, сигнальный сервер должен сообщить через ICE (или скриптом изменять на клиенте) только TCP/TLS варианты соединений на порты 80/443, необходимо использовать TCP прокси с анализом источника, он должен перенаправлять трафик с конкретного клиента на созданный порт TCP TURN для этого клиента, например Haproxy умеет так делать [6]. Это очень затратный способ и он обеспечит худшее качество, но связь будет работать.

Если запрещен только UDP трафик, то на TURN сервере для таких клиентов нужно принудительно оставлять только TCP , о чем сообщать через ICE (или скриптом изменять это на клиенте).

На многих брандмауэрах в NAT таблице небольшой таймаут для запоминания соединений, при этом если долго нет пакетов соединение удаляется и обратные пакеты уже не приходят. В этом случае нужен программный keep-alive в виде запросов по имеющимся открытым каналам данных пустых пакетов (программный Long-poll).

Проблемы с качеством изображения и звука более сложны в исправлении, поскольку иногда не имеют объективных показателей и метрик.

Обеспечение качества во время сеанса

Во время сеанса периодически (по набору показателей getStats() стека webRTC или просто по таймеру) начинается анализ части видеопотока. Поток отключается от непосредственной передачи, прореживается, перенаправляется в Canvas, там ему добавляется Watermark, рассчитывается перцептуальный хэш (п-хеш) с уменьшением изображения до 64x64, также слепок кадра сохраняется в base64 через функцию toDataURL(). Отрезок для определения качества не может быть менее 2-х секунд, он должен рассчитываться исходя из интервалов ключевых кадров (для VP8 он составляет 3000 кадров, но можно установить другой через параметр VP8KeyFrameInterval при установлении соединения, для VP9 этот интервал 9999 кадров), количества NACK сообщений.

После промаркированные фреймы перенаправляются в прежний поток вывода, а их п-хеши и копии в формате base64 отсылаются на сервер по другим каналам (или Data-channel WebRTC или POST/Websocket).

На приемной стороне происходит тоже самое с этим же потоком. После этого на сервере вычисляется расстояние Хемминга для изображений. Основной проблемой является синхронизация начала последовательности и возможные пропуски кадров в последовательности. Для этого на этапе маркировки начало последовательности выделяется сильнее чем остальные карды, потом ищутся наиболее близкие начальные кадры в обеих последовательностях [7]. Если совпадений фрагментов не найдено, то значит или нет видео, или оно сильно искажено (артефакты, замирание, низкая частота кадров, сильно уменьшенный битрейт). Этот способ подходит для периодического контроля и занимает довольно много времени и ресурсов. Кроме того, задержки видео могут составлять до нескольких секунд на спутниковых и сотовых каналах связи.

Если требуется осуществлять текущий контроль изображения, то используется тот же метод маркирования кадров, но частота прореживания на маркировку снижается до 1 кадра в 2 секунды. Хотя это и позволяет отследить только один кадр в 2 секунды, но также позволяет пол-

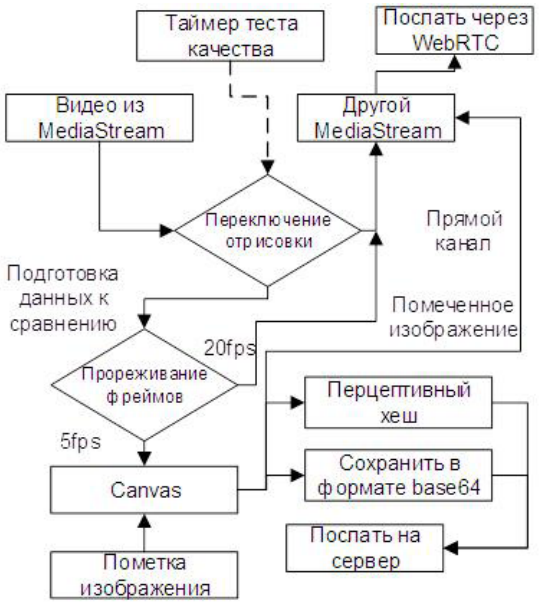

Рисунок 1 – Подготовка видео

ностью рассчитать расстояние Хемминга для этого кадра на обеих сторонах. Для более полного определения мест возникновения проблемы требуется этот кадр перехватить и хешировать в потоке RTP на сервере SFU, тогда будет ясно, на каком из плеч возникают проблемы.

Также существуют методы контроля не всего изображения, а только поиска лица и анализа его подвижности (face-api.js, picojs, trackingjs and etc.). С одной стороны этот метод позволяет на клиенте узнать о содержимом изображения, найти на нем лицо, понять, что изображение не замерло. С другой стороны он бесполезен при презентациях, вещании видео, потребление процессорного времени существенно возрастает.

Анализ голоса более сложен из за повышенных требований к минимизации задержек, возможной эхо-компенсации, а также из-за более сложной маркировки фрагментов. Любая заметная анализатору метка в звуковом потоке в доступных, но неслышимых частотах (при частоте семплинга WebRTC по умолчанию в кодеке Vorbis 44,1kHz, в кодеке OPUS 48кГц – менее 20Гц и более 20кГц) может исказить звук при неблагоприятном стечении обстоятельств. Кроме того в звуке нет фреймов и анализировать нужно только фрагмент звуковой дорожки. Для постоянного использования нужно учесть тот момент, что звук частот ниже 20Гц и выше 20кГц может возникать одновременно при стуках и ударах по микрофону, поэтому в схему добавлен предварительный полосной фильтр.

На удаленной стороне на входе потока должен стоять спектральный анализатор для выявления несущих низких и высоких частот, а также полосной фильтр для очистки от них исходного звука. При выявлении этих частот можно начинать запись п-хеша фрагмента для анализа [8].

Напрямую несущие частоты в анализе использовать нельзя, поскольку они сильно искажаются как фильтрами, так и кодеком и наложение других звуков, могут прерываться из-за потери пакетов.

На сервере анализ аудиодорожки происходит по другому, чем с в видео. Поскольку дорожка с удаленной стороны может быть подмножеством дорожки с исходной стороны (непрерывная часть, множество отрезков), то требуется провести анализ как в целом (через п-хеш), так и отдельных частей. Для этого и посылается записанная дорожка целиком на сервер при необходимости. Если анализ по п-хешу выявил существенные различия, требуется сначала выявить возможные провалы в звуке, вызванные потерями пакетов. Для этого сравниваются громкости двух дорожек, выявляется возможное начало и конец совпадающих фрагментов, а также участки с несовпадающими данными. На несовпадающих участках производится спектральный анализ, если спектр различен в пределах заданных погрешностей, то считаем, что это аномалия и помечаем участок как испорченный.

Автоматизированное тестирование в эксперименте

Для проведения эксперимента был использован фреймворк Selenium, который позволяет загружать много копий браузера и автоматизировать действия с ними. Для работы с браузерами и написания скриптов тестирования использовался KITE [9] , в котором есть примеры под популярные сигнальные серверы, а также для P2P связи WebRTC. Во фреймворке используется модуль kite-common, в котором есть небольшая часть по определению наличия видео и аудио. Видео определяете по копии кадра, в котором рассчитывается сумма цветов всех точек. Если

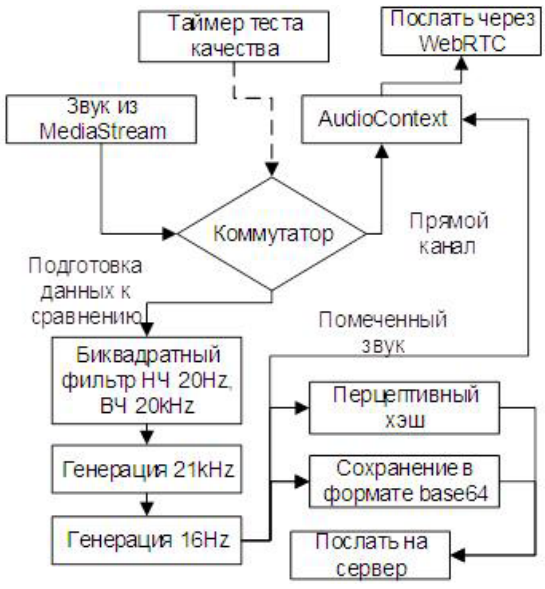

Рисунок 2 – Подготовка видео

кадр черный, то сумма будет около нуля, если с белой заливкой – то как максимальная сумма при данном размере кадра. Для определения качества и наличия артефактов такой метод не подходит, поэтому был реализован метод, описанный в главе IV. Контроль звука осуществлялся по наличию несущей. Для генерации видео источника в оригинальном kate используется ключ запуска «--use-fake-device-for-media-stream», что позволяет посылать картинку с временем и крутящимся сектором, а также со звуком (регулярный beep). Но для проверки QoE этого недостаточно, были использованы дополнительные опции «--use-file-for-fake-audio-capture=file.y4m --use-file-for-fake-video-capture=file.wav --allow-file-access», которые позволили воспроизвести реальное видео и звук. Видео и звук были записаны на реальной веб-конференции и имели размеченные заранее проблемы с периодически крутящейся камерой (для резких перепадов изображений) и с громкими звуками (писк от связи микрофона и динамика, падение микрофона на стол),

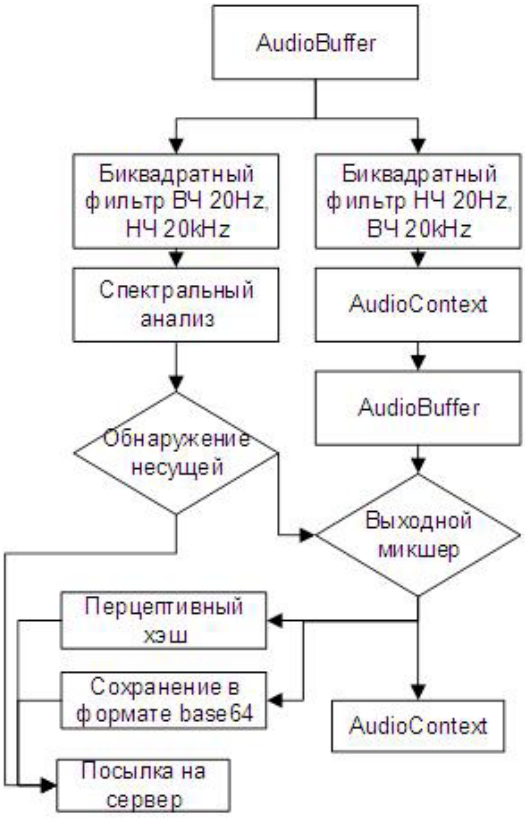

Рисунок 3 – Анализ звука

Эксперимент

Тестирование проводилось на поднятой в облаке инфраструктуре с 2 виртуальными машинами, на первой установлен в качестве сигнального сервера Janus, он настроен без использования авторизации на порту 8089 по протоколу websocket, для защищенного канала использовались сертификаты LetsEncrypt. Также был создан простой скрипт на PHP для генерации логинов и паролей TURN сервера и взаимодействия с API coturn. Второй сервер использовался как TURN сервер, на нем был установлен coturn с авторизацией по логину/паролю и ключу, включенным API, SSL/TLS шифрованием и сертификатом LetsEncrypt.

В первой части эксперимента мы проанализируем нагрузку на процессор и память различных методов работы с видео и аудио источниками: распознавание лица, маркирование видео, сохранение видео в base64, маркирование аудио, анализ голосовых частот, анализ наличия несущей. Во второй части будем испытывать все нужные элементы вместе. Здесь и далее загрузка процессора будет дана общая на браузер, без разбивки на составляющие.

Проверка определения лица проводилась на браузере Firefox ESR 52 в Windows 10 Pro 1903, компьютер Core-i7-3517/ 8Gb RAM/ SSD. Chrome также был испытан, но не все пример, которые запустились на Chrome без ошибок запустились на Firefox, кроме того загрузка процессора на Chrome . С библиотекой fase-api.js [10] получены следующие результаты:

1 Face Detection remote video

SSD Mobilenet v1 – нагрузка 30% CPU, таймаут определения лица 960ms (1 fps);

Tiny Face Detector – нагрузка 28% CPU, таймаут определения лица 160 msec (6 fps);

2 Face Detection local webcam

SSD Mobilenet v1 – нагрузка 36% CPU, таймаут определения лица 550ms (1.7 fps);

Tiny Face Detector – нагрузка 38% CPU, таймаут определения лица 55 msec (18 fps).

Были протестированы и другие библиотеки:

- picojs [11], работает только при прямом взгляде камеру, максимальный наклон головы 5%, нагрузка CPU 38%, таймаут определения лица около 500мс)

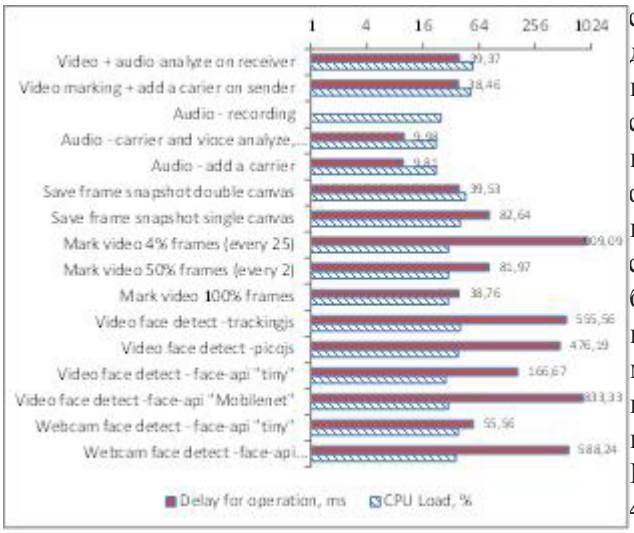

Рисунок 4 – Сравнение производительности методов

- trackingjs [12], не поддерживает поворот головы и наклон более 15%, нагрузка 40% CPU, таймаут определения лица около 600мс.

В реальном видео на компьютере в браузере определения лица пока ненадежно и очень ресурсоемко.

Анализ маркирования видео проводился на том же оборудовании по примеру [13].

Маркировка всего потока 25fps – нагрузка 30% CPU, маркировка половины потока 12 fps, нагрузка 30% CPU; маркировка каждого 25 кадра – , нагрузка 30% CPU. Во всех случаях по показателям внутренней нагрузки браузера, до 90% всей нагрузки занимал процесс GPU.

Сохранение в base64 всего видео потока за период времени – загрузка 40% CPU, среднее время на 1 кадр – 86 мс, поддерживаемая частота кадров– 12 fps; при использовании предварительного копирования кадров загрузка 46% CPU и частота (с быстрым копированием кадра в два скрытых Canvas) – 25 fps.

Генерация несущей и добавление в аудиопоток – загрузка 22% CPU, задержка менее 10мс.

Анализ несущей, голосовых частот и применение эквалайзера в аудиопоток – загрузка 22% CPU, задержка менее 10мс.

Запись звука – нагрузка 25% CPU.

Теперь проверим, как поведет себя браузер, когда все элементы соберутся вместе в единую систему генерации и проверки. Для этого в видео на источнике добавляется 1 метка в углу каждые 25 кадров, этот кадр посылается на сервер. Был создан прототип системы проверки. На приемнике сначала идет анализ заранее определенной части в углу кадра всех кадров в для поиска характерной метки для начала синхронизации потока. Метка должна быть небольшой (не более 64x64) и контрастной, поиск идет по сравнению с присланным п-хешем метки (функция phash требует 2.7 мс начиная с копирования и масштабирования Canvas и заканчивая вычислением расстояния Хемминга). Весь процесс поиска не должен быть длиннее 40мс (время кадра), с учетом прорисовки основного Canvas (около 3мс)

Загрузка как на передающем, так и на принимающем компьютере была около 50% CPU (на браузере Chrome – до 70%). Наличие видео метки детектируется не дальше 2-3 кадра от начала вещания, наличие голосовой метки – около 0.5-1 секунды, максимальная длительность одного цикла подготовки видео и аудио при формировании посылки – 38мс, при анализе на приемной стороне – 39мс.

Как видно из графика, что существуют операции, которые длиннее 40 мс и не могут быть выполнены в реальном времени на видеопотоке 25 FPS.

Выводы

В результате исследования были рассмотрены методы контроля и обеспечения качества связи по протоколам WebRTC, рассмотрены методы контроля на клиентской стороне, испытан прототип системы контроля. Прототип системы контроля показал задержку на кадр в районе 38-39мс, что позволяет в реальном времени отслеживать отличия в видеопотоке от оригинала и реагировать на это.

В дальнейших исследованиях будет уделено внимание точности отслеживания ухудшения качества изображения и звука, исследовано влияние кодеков и их параметров на точность определения отличий, рассмотрена автонастройка параметров кодеков и протоколов.

14.02.2019

Работа выполнена при поддержке Российского фонда фундаментальных исследований в рамках проектов 19-47-560011, 18-37-00485, 18-07-01446 и гранта Оренбургской области 2019 г. № 32.

Список литературы:
1. Digital Voice Ports, or work with ISDN E1 PRI [Электронный ресурс]. – CiscoMaster, 2013. Режим доступа: http://ciscomaster.ru/content/digital-voice-ports-ili-rabota-s-isdn-e1-pri.
2. WebRTC Troubleshooter [Электронный ресурс]. – WebRTC, 2017. Режим доступа: https://test.webrtc.org/.
3. Check WebRTC STUN and TURN connectivity [Электронный ресурс]. – GitHub, 2014. Режим доступа: https://github.com/otalk/stunturncheck.
4. Simple WebRTC video/voice and data channels [Электронный ресурс]. – GitHub, 2019. Режим доступа: https://github.com/feross/simple-peer.
5. Hector Zelaya. Effects In WebRTC? A Filters Tutoria [Электронный ресурс]. – WebRTC Ventures, 2018. Режим доступа: https://webrtc.ventures/2018/12/effects-in-webrtc-a-filters-tutorial/.
6. Running HTTPS, SSH and VPN on port 443 [Электронный ресурс]. – Limbenjamin, 2016. Режим доступа: https://limbenjamin.com/articles/running-https-ssh-vpn-on-port-443.html.
7. Ola Kjelsrud. Using Perceptual Hash Algorithms to Identify Fragmented and Transformed Video Files [Электронный ресурс]. – Gjovik University College, 2014. Режим доступа: https://pdfs.semanticscholar.org/8285/4824363b4088cc65d49da7f7d8bea5b8082c.pdf.
8. Shut up! Monitoring audio volume in getUserMedia [Электронный ресурс]. – WebRTC Chacks, 2017. Режим доступа: https://webrtchacks.com/getusermedia-volume/.
9. KITE is a test engine designed to test WebRTC interoperability across browsers [Электронный ресурс]. – GitHub, 2019. Режим доступа: https://github.com/webrtc/KITE.
10. JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js [Электронный ресурс]. – GitHub, 2019. Режим доступа: https://github.com/justadudewhohacks/face-api.js/.
11. A face detection library in 200 lines of JavaScript [Электронный ресурс]. – GitHub, 2019. Режим доступа: https://github.com/tehnokv/picojs.
12. A modern approach for Computer Vision on the web [Электронный ресурс]. – Режим доступа: https://trackingjs.com.
13. Client side WebRTC lib to add an image / watermark on the MediaSource [Электронный ресурс]. – GitHub, 2017. Режим доступа: https://github.com/asafrob/WebRtcShitBlt.

References:
1. Digital Voice Ports, or work with ISDN E1 PRI. CiscoMaster, 2013. Available at: http://ciscomaster.ru/content/digital-voice-ports-ili-rabota-s-isdn-e1-pri.
2. WebRTC Troubleshooter. WebRTC, 2017. Available at: https://test.webrtc.org/.
3. Check WebRTC STUN and TURN connectivity. GitHub, 2014. Available at: https://github.com/otalk/stunturncheck.
4. Simple WebRTC video/voice and data channels. GitHub, 2019. Available at: https://github.com/feross/simple-peer.
5. Hector Zelaya. Effects In WebRTC? A Filters Tutoria. WebRTC Ventures, 2018. Available at: https://webrtc.ventures/2018/12/effects-in-webrtc-a-filters-tutorial/.
6. Running HTTPS, SSH and VPN on port 443. Limbenjamin, 2016. Available at: https://limbenjamin.com/articles/running-https-ssh-vpn-on-port-443.html.
7. Ola Kjelsrud. Using Perceptual Hash Algorithms to Identify Fragmented and Transformed Video Files. Gjovik University College, 2014. Available at: https://pdfs.semanticscholar.org/8285/4824363b4088cc65d49da7f7d8bea5b8082c.pdf.
8. Shut up! Monitoring audio volume in getUserMedia. WebRTC Chacks, 2017. Available at: https://webrtchacks.com/getusermedia-volume/.
9. KITE is a test engine designed to test WebRTC interoperability across browsers. GitHub, 2019. Available at: https://github.com/webrtc/KITE.
10. JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js. GitHub, 2019. Available at: https://github.com/justadudewhohacks/face-api.js/.
11. A face detection library in 200 lines of JavaScript. GitHub, 2019. Available at: https://github.com/tehnokv/picojs.
12. A modern approach for Computer Vision on the web. Available at: https://trackingjs.com.
13. Client side WebRTC lib to add an image / watermark on the MediaSource. GitHub, 2017. Available at: https://github.com/asafrob/WebRtcShitBlt.

Сведения об авторах:

Ушаков Юрий Александрович, доцент кафедры геометрии и компьютерных наук
Оренбургского государственного университета, кандидат технических наук
E-mail: unpk@mail.ru; ORCID 0000-0002-0474-8919
Шухман Александр Евгеньевич, заведующий кафедры геометрии и компьютерных наук
Оренбургского государственного университета, кандидат педагогических наук, доцент
E-mail: shukhman@gmail.com; ORCID 0000-0002-4303-2550
Парфенов Денис Игоревич, доцент кафедры прикладной математики
Оренбургского государственного университета, кандидат технических наук
E-mail: parfenov_di@mail.ru; ORCID 0000-0002-4303-2550
Полежаев Петр Николаевич, старший преподаватель кафедры компьютерной безопасности и математического обеспечения информационных систем Оренбургского государственного университета
E-mail: newblackpit@mail.ru; ORCID 0000-0002-4303-2550
Ушакова Маргарита Викторовна, старший преподаватель кафедры геометрии и компьютерных наук
Оренбургского государственного университета
E-mail: m.v.ushakova@mail.ru; ORCID 0000-0002-4303-2550
460018, г. Оренбург, пр-т Победы, д. 13