В.А. Ткаченко. **МЕТОДИКИ РАЗРАБОТКИ WEB-КОММУНИКАЦИОННЫХ СЕРВИСОВ РЕАЛЬНОГО ВРЕМЕНИ** / В.А. Ткаченко, В. А. Рябик // Кафедра систем інформації: Зб. наук. праць / Під ред. проф. Кравця В.О. та проф. Серкова О.А. – Х.: ТОВ «Щедра садиба плюс», 2014. – С. 80-97.

Предлагаются методики разработки Web коммуникационных услуг в режиме реального времени. WebRTC с WebSocket, HTML5, CSS3 и JavaScript позволяют создавать веб-коммуникационные сервисы нового поколения. Эти сервисы для общения через Интернет используют только веб-браузеры без плагинов и дополнительного программного обеспечения на оконечных устройствах пользователя.

Methodologies of development of Web of communication services are offered real-time. WebRTC with WebSocket, HTML5, CSS3 and JavaScript allow to create of web-communication services of new generation. These services for communication over the Internet use only web-browsers without plugins and additional software on the eventual devices of user

Введение. В настоящее время наряду с традиционными телекоммуникациями активно применяются интернет-коммуникации, и наблюдается их конвергенция. Принцип конвергенция всех служб связи реализуется в сетях связи NGN. Базовой основой NGN являются опорные IP-сети, которые могут обеспечить интеграцию услуг передачи всех типов трафика: данных, голоса и мультимедиа. В сети NGN, которая должна обеспечить передачу все типов трафика, основными слоями являются транспортный и сервисный. Если в эволюции транспортных сетей, наблюдается формирования конвергентного сетевого уровня P-OTN, то в эволюции услуг связи (сервисных услуг), наблюдается формирование конвергентных веб-коммуникационных услуг в режиме реального времени.

Интернет-коммуникации - это способы общения людей через сеть Интернет. Наиболее перспективными направлениями Интернет-коммуникаций являются Web - коммуникации в режиме реального времени. Web-коммуникации в режиме реального времени - это способы общения через IP-сети с помощью веб-браузера без установки плагинов и расширений. Web-коммуникации в режиме реального времени - это сервисы, построенные по архитектуре "клиент-клиент".

Анализ основных достижений и литературы. К интернет-коммуникационным сервисам относятся: системы обмена сообщениями, on-line video и VoIP. Службы обмена сообщениями делятся на службы обмена сообщениями в режиме off-line (E-mail, рассылка SMS) и службы мгновенных сообщений (IRC, веб чаты, IM, PTT и так далее) в режиме on-line.

Чаты, веб чаты, голосовые и видео чаты в веб интерфейсе, IM, VoIP, - это сервисы, которые обеспечивают интернет-коммуникации в режиме онлайн через составные сети с пакетной коммутацией. Следует отметить, что системы обмена сообщениями имеют свои коммуникационные сети, и в основном построены по архитектуре "клиент-сервер". Интернет-коммуникационные сервисы требуют или установки клиентских приложений на пользовательские устройства, или установки плагинов и расширений в веб-браузеры.

Сервисы, которые обеспечивают интернет-коммуникации в режиме онлайн, являются приложениями, в которых каналы передачи голоса, видео, данных, текста и файлов, как правило, не интегрированы, за исключением некоторых IM, например Skype [1]. Некоторые из систем связи работают на частных (закрытых) протоколах, например Skype.

Как правило, указанные приложения не могут одновременно работать в нескольких коммуникационных сетях, т.е. системы связи не могут взаимодействовать друг с другом, в результате чего необходимо устанавливать отдельные приложения для каждого сервиса. Но следует отметить, что некоторые системы связи открывают спецификации своих протоколов и пытаются предоставить возможность общения между разными системами. Кроме того, применяется Gtalk2voip - сервис, предоставляющий шлюз для голосового взаимодействия с пользователями других сетей. Применяются многопротокольные IM-клиенты, которые могут работать в нескольких сетях одновременно без использования шлюзов [2]. Но этого явно недостаточно для решения проблемы унификации систем связи.

На основании вышеизложенного материала основными проблемами конвергенции интернет-коммуникационных услуг является:

- интеграция различных видов связи в одном сервисе;
- обеспечение синхронизации различных видов связи между собой и взаимодействия их со всеми существующими сетями (PSTN, SIP, LTE, IMS);
- применение открытых протоколов связи в коммуникационных услугах реального времени;
- применение веб-браузеров (интегрированных клиентов) в качестве единых средств общения (интерфейсов) различных терминальных устройств.

Проблему конвергенции коммуникационных услуг реального времени, т.е. интеграцию каналов передачи голоса, видео, данных и доступ к ним с помощью единого сетевого приложения (веб-браузера) можно решить на основе технологий WebRTC, HTML5, CSS3, протоколов WebSocket и приложений SIPML5, webrtc2sip [3, 4, 5, 6, 7, 8].

По сути, браузеры, поддерживающие WebRTC, WebSocket и SIPML5 могут быть единым средством (интерфейсом) для всех пользовательских устройств (ПК, смартфонов, iPad, IP-телефонов, мобильных телефонов и т.д.), которые обеспечивают Web- коммуникации в реальном времени [9].

В настоящее время большинство веб чатов, как способов коммуникаций через Интернет, основаны на сетевой клиент-серверной архитектуре. Роль клиентской части веб чатов выполняет браузер.

Но на смену клиент-серверной архитектуре, приходят одноранговые или пиринговые архитектуры сети (P2P, point - to - point), которые могут решить проблему интеграции каналов передачи голоса, видео и данных в одном устройстве - веб-браузере.

В основном для создания P2P видеочатов применяются flash технологии с низким качеством передачи мультимедийных данных. Кроме того, для вывода голоса и видеопотока из микрофона и видеокамеры в p2p flash видеочатах необходима установка flash плагина в веб-браузер.

Одним из современных решений в сфере веб-коммуникаций являются технологии WebRTC, HTML5 с JavaScript и CSS3, а также протокол WebSocket, предусмотренный в спецификации HTML5, с помощью которых решаются проблемы создания P2P видеочата без применения плагинов [10]. Кроме того, WebRTC обеспечивает лучшее качество передачи звука и видео чем Flash.

WebRTC: это открытая технология, предназначенная для создания пиринговых сетей связи, которая позволяет пересылать текстовые и мультимедийные данные непосредственно между браузерами без помощи сервера

Сигнальный сервер используется только для установки p2p соединения между двумя браузерами или WebRTC клиентами. Программный код клиентской и серверной части P2P чата реализуется на JavaScript. Клиентское приложение взаимодействует с браузерами через API WebRTC [11].

Технология WebRTC реализуется тремя интерфейсами JavaScript API:

- RTCPeerConnection - для создания соединения "клиент-клиент" или Peer-to-Peer между браузерами (реализует аудио/видео звонки);
- Media Stream (getUserMedia) - для захвата микрофона, видео камеры и передачи мультимедиа;
- RTCDataChannel - для передачи данных.

В спецификации HTML5, предусмотрены такие новые элементы (теги) как canvas, video, audio, которые управляются JavaScript и предназначены для вывода голоса и видео потока с микрофона и видеокамеры пользователя.

Спецификация WebSocket представляет интерфейс JavaScript API, который определяет полнодуплексное сокетное соединение (коммуникацию в обоих направлениях одновременно), по которому сообщения могут быть отправлены между клиентом и сервером. WebSocket – обеспечивает полностью асинхронный и симметричный обмен сообщениями между браузером и сервером через один TCP-сокет. В некоторых случаях в P2P чатах для асинхронного обмена сообщениями между браузером и сервером может быть применен Ajax (XMLHTTPRequest).

Приложение SIPML5 для VoIP или SIP клиента для браузера, написанное на HTML/CSS/JavaScript, и шлюз webrtc2sip открыли путь к созданию SIP-софтфона в web интерфейсе или SIP клиента в веб-браузере. Теперь из любого веб-браузера, поддерживающего WebRTC, с помощью SIPML5 можно совершать звонки или видеозвонки как на сети ТфОП, так на сети: SIP, LTE, IMS через шлюз webrtc2sip или сервер Asterisk (версия 11 и выше).

Цель исследования. Цель этой работы создать методики разработки Web-коммуникационных сервисов в реальном времени на основе современных технологий, протоколов и приложений, исключив необходимость установки дополнительных программных продуктов и плагинов в устройствах пользователей.

Постановка задачи. Разработать методики создания Web-приложений реального времени или Web - коммуникационных сервисов в реальном времени: p2p видеочатов и SIPml5 - софтфонов в web интерфейсе на основе HTML5 и WebRTC, которые обеспечивают текстовое, голосовое и видео общения пользователей через Интернет с использованием только веб-браузеров без установленных у них дополнительных программ.

Материалы и результаты исследований. В предлагаемой методике разработки p2p видеочата была применена технология WebRTC с использованием HTML5, CSS3 и WebSocket. В общем случае основными компонентами схемы взаимодействия p2p видеочата являются:

- веб-браузеры, которые поддерживают спецификацию HTML5 WebSocket, WebRTC;
- сигнальный сервер;

- сервер для обхода NAT трансляторов;
- сервер аутентификации.

Схема взаимодействия компонентов p2p видеочата представлена на рисунке 1.

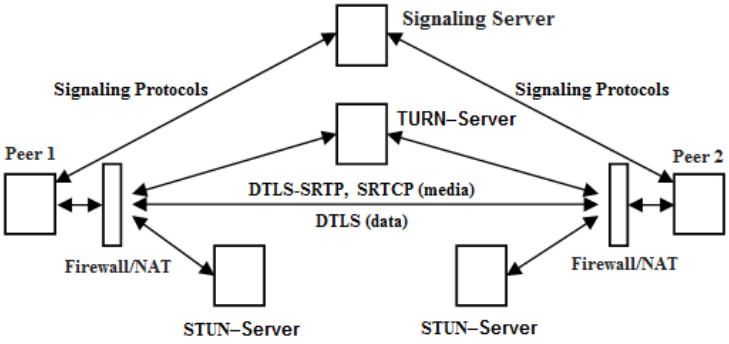

Рисунок 1 - Схема взаимодействия компонентов p2p видеочата

В процессе разработки p2p видеочата необходимо решить несколько задач:

- определиться с типом сигнального сервера (сервера согласования сеанса связи);
- выбрать STUN или TURN серверы, доступные для обхода NAT транслятора, если клиенты находятся за NAT;
- выбрать способ авторизации;
- выбрать среду разработки;
- выбрать средства реализации (библиотеку или API) кода клиентской и серверной частей;
- создать веб-приложение.

Определение типа сигнального сервера. Связь браузеров координируются через канал сигнализации или канал согласования сеанса. Стандарт WebRTC не устанавливает способы реализации канала сигнализации. В качестве сервера согласования сеанса могут быть использованы HTTP/HTTPS – server, WebSocket-server или XMPP-server. В качестве протоколов обмена сообщениями между браузером и HTTP/HTTPS сервером – Ajax или WebSocket, а в качестве сигнальных протоколов для XMPP-сервера - XMPP/WebSocket или Jingle/XMPP [12, 13].

Применение протоколов Ajax или WebSocket для взаимодействия браузеров и сервера обусловлено тем, что после установления связи между клиентами связь клиентов с сервером должна поддерживаться по протоколу Ajax или WebSocket для обмена информацией, например данными о присутствии в сети пользователей из списка контактов. Ajax и WebSocket обеспечивают дуплексную связь между клиентским и серверным приложениями, т.е. создают канал согласования сеанса. Браузер и клиентское веб-приложение JS/HTML/CSS взаимодействуют через WebRTC API. Схема канала согласования сеанса представлена на рисунке 2.

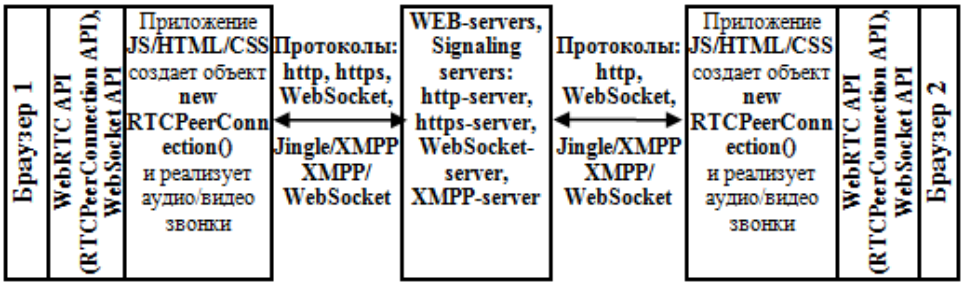

Рисунок 2 – Схема канала сигнализации или канала согласования сеанса

Следует отметить, что сигнализация не является частью RTCPeerConnection, но обмен сигнальными сообщениями между пирами выполняется сигнальными протоколами через сервер в процессе создания PeerConnection.

После обмена сигнальными сообщениями между браузерами, WebRTCPeerConnection создает между ними дуплексную связь для обмена видеопотоками и произвольными данными. Схема канала мультимедийных потоков и данных представлена на рисунке 3.

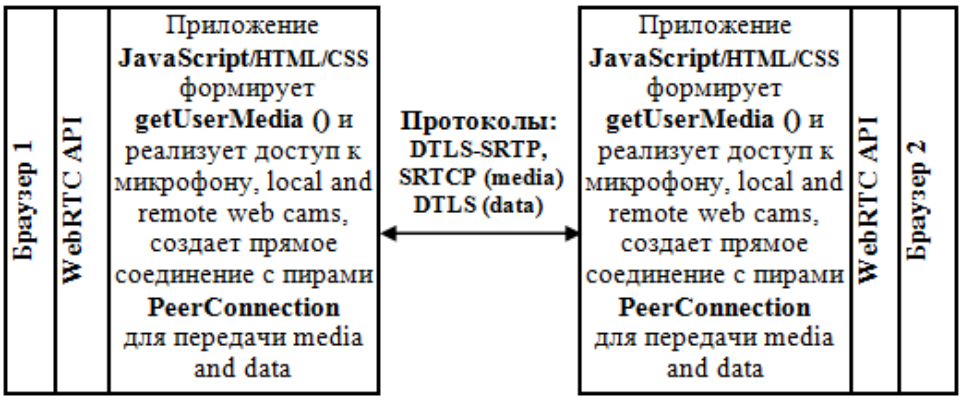

Рисунок 3 - Схема канала мультимедийных потоков и данных

WebRTC обеспечивает безопасное зашифрованное соединение браузеров по одноранговой схеме. Браузеры передают мультимедийные потоки по протоколам DTLS-SRTP, а произвольные данные по протоколу DTLS, которые работают поверх UDP.

Выбор STUN или TURN серверов. Поскольку брандмауэры и NAT создают проблемы для браузеров (клиентов), находящихся за брандмауэрами или NAT, которые используют p2p соединения через сеть Интернет, то для обхода брандмауэров и NAT трансляторов используются серверы STUN и TURN. STUN и TURN – это с одной стороны серверы, с другой - прикладные протоколы.

С помощью STUN- сервера клиентское приложение (рис. 4) определяет истинный IP-адрес клиента, если он находится за NAT. STUN-сервер и клиентское приложения взаимодействуют с помощью протокола STUN. STUN - это клиент-серверный протокол, который работает поверх транспортного протокола UDP.

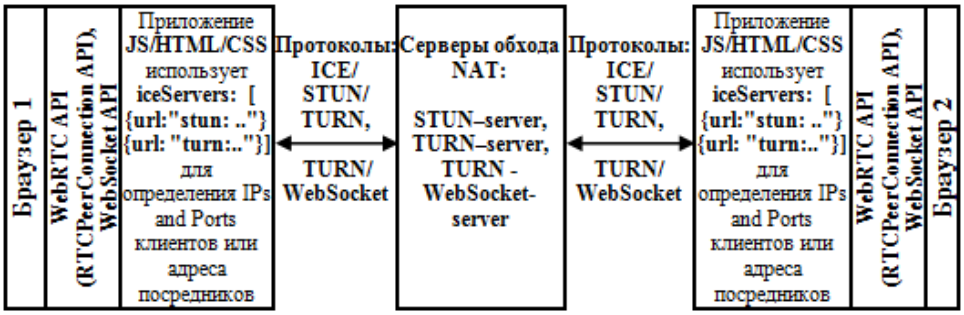

Рисунок 4 - Канал обхода NAT и брандмауэров

В P2P чатах, как правило, применяется публичный STUN-сервер, например stun:stun.l.google.com:19302. Для установления связи партнеры должны обменяться

ICE-candidate и описанием (мультимедийного контента) сеансов SDP через сигнальный сервер. Полученная с STUN-сервера информация об IP-адресе и номере порта клиента (ICE- candidate: IP and Port) передается партнеру через сигнальный сервер. Клиентские приложения, зная адреса ICE- candidates и описание сеансов SDP, могут установить UDP-соединения для передачи мультимедийных потоков и произвольных данных. Необходимо отметить, что к ICE- candidate относятся любые IP-адреса (истинные адреса клиентов за NAT или адреса посредников) по которым могут быть доступны клиенты.

Когда STUN не может быть использован для определения ICE-candidate, то применяется сервер посредник или TURN-сервер. Для общения с TURN-сервером применяется клиент-серверный протокол TURN, который работает поверх TCP. TURN-сервер или Relay server возвращает транспортный адрес посредника (IP и номер порта сформированные TURN-сервером) запросившему клиенту, который он передает партнеру через сигнальный протокол. Теперь клиент может получить информацию от партнера, который посылает данные через TURN-сервер, т.е. путем ретрансляции данных через TURN-сервер, который расположен в сети Интернет. Таким образом, формируются маршруты с ретрансляцией медиатрафика через TURN-сервер. Например, для обхода NAT используется TURN-сервер http://numb.viagenie.ca. Поскольку весь медиатрафик и трафик данных проксируются через TURN-сервер, то это метод применяется в исключительных случаях.

Следует отметить, что, если клиенты не находятся за брандмауэрами и NAT, то в приложениях не используется функция обхода NAT. Для определения приоритета обхода NAT применяется протокол ICE (интерактивное создание подключений), который базируется на протоколах STUN и TURN [14]. ICE устанавливает приоритеты от высшего к низшему в такой последовательности: без применения STUN, с применением STUN, с применением TURN-сервер.

Выбор способа авторизации. Как правило, p2p видеочаты являются защищенными ресурсами и для общения в p2p видеочате требуется зарегистрироваться в этом сервисе. Авторизоваться в p2p видеочате можно через серверы социальных сетей (серверы аутентификации), если открыть p2p приложению доступ к учётной записи в социальной сети. Для этого используется протокол OAuth 2.0 [15], который позволяет привлекать посторонний сервис (например, социальные сети) для авторизации без передачи логина и пароля p2p видеочату. OAuth 2.0 основан на GET и POST запросах и редиректах. OAuth оперирует с тремя объектами: клиент, веб-приложение (ресурс) и владелец ресурса. Чтобы применить эту технологию владелец созданного веб-приложения должен зарегистрировать это веб-приложение в соцсети, например в Facebook по адресу https://developers.facebook.com/apps. Полученный код кнопки «Войти через Facebook», владелец ресурса добавляет в код страницы регистрации видеочата, через эту кнопку можно войти на созданный сервис (p2p видеочат). Для входа в веб-приложение клиент (пользователь зарегистрированный в Facebook) должен открыть страницу регистрации и щелкнуть на кнопке «Войти через Facebook». Открывается окно авторизации («Войдите в Facebook, чтобы использовать свой аккаунт в приложении p2p видеочат»). В открывшемся окне, пользователь должен ввести свой логин и пароль. В случае успешной авторизации производится обмен GET или POST запросами между веб-приложением и Facebook. Facebook выдает клиенту Key + Secret (набор символов), который называется токен, а браузер редиректится на

URL веб-приложения. Предъявив токен веб-приложению, клиент получает доступ к защищенным ресурсам p2p видеочата. Следует отметить, что клиенту предоставляются ограниченные возможности ресурса. API социальных сетей предоставляет возможность подключать внешние веб приложения или веб-сайты, используя OAuth-аутентификацию.

Выбор среды разработки. Разработку Web-коммуникационных сервисов в режиме реального времени можно выполнять в файловой системе ПК или на облачном сервисе. Установка node.js в файловую систему ПК выполняется следующим образом. Сначала надо открыть страничку http://nodejs.org/download/. Далее для пользователей windows скачать win.installer (.msi), запустить win.installer (.msi) на ПК и установить nodejs, а также npm package manager в директорию Program Files. Диспетчер npm необходим для добавления модулей в директорию разрабатываемых веб-приложений. Для испытаний в реальной среде можно развернуть пиринговый видеочат на хостинге с поддержкой Node.js https://www.dotcloud.com/ или на облачном сервисе https://www.appfog.com/.

Выбор средств реализации кода клиентской и серверной частей. Разработку Web-коммуникационных сервисов в реальном времени можно вести на основе набора API-интерфейсов WebRTC, а можно на основе набора API-интерфейсов и готовых библиотек, например webrtc.io. Исходники доступны в репозитории GitHub. GitHub - это хранилище библиотек открытого, свободного исходного кода. GitHub — это сайт или веб-сервис для совместной разработки и хранения исходного кода.

Методика разработки p2p видеочата:

1) установить на ПК среду разработки и выполнения JavaScript кода вне браузера node.js;

2) создать директорию для веб-приложения на диске C;

3) в созданную директорию добавить необходимые модули;

4) создать код серверной части коммуникационного приложения на JS;

5) создать код клиентской части коммуникационного приложения на JS;

6) разработать интерфейс клиентского приложения с помощью гипертекстовой разметки HTML5 и CSS3.

После установки node.js на ПК создаем директорию, например p2p, на диске C. В созданную директорию добавляем необходимые для разработки модули. Для установки модулей необходимо в командной строке из директории приложения (например, "p2p") выполнить команду: npm install имя_модуля. В процессе установки модулей npm-менеджер создает папку node_modules в директории, из которой была выполнена установка. В процессе работы nodejs автоматически подключает модули из директории node_modules. Для ускоренной разработки веб-приложений целесообразно подключить модуль express. Модуль express - это фреймворк для node.js или веб-платформа для разработки приложений. Итак, в командной строке выполняем C:\p2p-chat >npm install express. При необходимости можно добавить модуль СУБД MySQL, набрав в командной строке C:\p2p>npm install MySQL, или xmpp сервер C:\p2p>npm install node-xmpp и т.д.

Реализация серверной части приложения server.js. Чтобы создать веб-сервер и WebSocket-сервер на базе NodeJS можно воспользоваться модулями http и webrtc.io [16]. Для этого добавим модуль webrtc.io в директорию p2p (C:\p2p>npm install webrtc.io), который содержит webrtc.io.js (SERVER). Если создаем видеочат с

применением базы данных, то в директорию p2p добавим модуль СУБД MySQL. Далее в директории p2p создаем файл server.js.

Серверная часть приложения server.js должна содержать строки:

var app = require ('express') () \\ подключаем фреймворк express

var server = require('http').createServer(app) \\ подключаем http-модуль, и создаем http server

var webRTC = require('webrtc.io').listen(8888) \\ подключаем модуль webrtc.io, и ставим на прослушивание порт, например 8888. Добавляем в серверный скрипт файлы, используя HTTP-метод фреймворк express для обеспечения API маршрутизации (index.html, style.css, script.js, webrtc.io.js).

Затем запускаем сервер командой: C:\ p2p-chat >node server.js. Если сервер работает успешно, приступаем к созданию клиентского приложения.

Реализация клиентской части коммуникационного приложения script.js. В папке p2p создаем клиентский script.js, применив API WebRTC, и поместим в папку p2p готовую библиотеку для клиентского приложения webrtc.io.js [17].

Клиентский script.js предназначен для управления приложением, он осуществляет обработку видео, отправку/прием текстовых сообщений, подключение комнат, управляет масштабом изображения (обычное, полноэкранное) и т.д.

Разработка интерфейса коммуникационного дополнения. Разработка каркаса интерфейса или веб-страницы index.html выполняется с помощью языка гипертекстовой разметки HTML5.

Файл index.html создаем в папке p2p. Форматирование html страницы осуществляется с помощью CSS3, файл CSS3 создаем в директории p2p. С помощью директивы <!DOCTYPE html> веб-страница объявляется как документ HTML5. Чтобы показать изображение с камеры в код веб-страницы надо добавить тег <video> с указанием параметров окна изображения и его размещения. Создаются элементы div с именами chatbox и chatinput для передачи текста, buttonBox (кнопки fullscreen и newRoom) для изменении масштаба и открытия новой комнаты общения. Объекты инициализируются из события onload тега body. С помощью тега <script> в html-код веб-страницы встраиваются клиентские скрипты: script.js и библиотека webrtc.io.js. Причем webrtc.io.js помещается в теги <head> веб-страницы.

SIP-софтфон в web интерфейсе на основе WebRTC, SIPML5, webrtc2sip.

В настоящее время существуют две демонстрационные версии софтфонов в браузере, поддерживающих WebRTC, - это SIPML5 и jsSIP. В предложенной методике разработки VoIP-софтфона в веб-браузере с аудио, видео и SMS возможностями была применена технология WebRTC с использованием HTML5, CSS3, WebSocket, клиентского приложения SIPML5 и шлюза webrtc2sip.

В общем случае основными компонентами схемы взаимодействия SIP-софтфона в web интерфейсе являются:

- веб-браузеры, которые поддерживают спецификацию HTML5, WebSocket, WebRTC и стек SIP/SDP;
- шлюз webrtc2sip;
- сигнальный сервер SIP Proxy;
- сервер для обхода NAT транслятора.

Схема взаимодействия компонентов двух VoIP-софтфонов в веб-браузере аналогична схеме взаимодействия компонентов p2p видеочата, представленной на

рисунке 1. В этом случае в качестве сигнального сервера применяется SIP Proxy. Если взаимодействуют VoIP-софтфон в веб-браузере и традиционный SIP телефон, который не поддерживает WebRTC, то схема их взаимодействия имеет вид, представленный на рисунке 5.

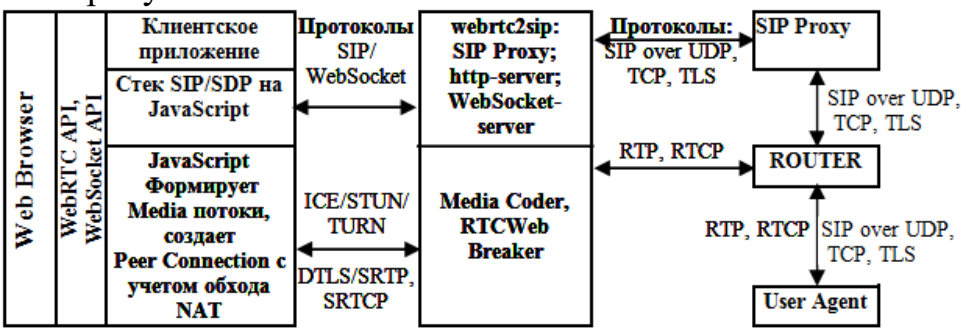

Рисунок 5 - Схема взаимодействия компонентов VoIP-софтфона в веб-браузере и традиционного SIP телефона (User Agent)

Из любого веб-браузера, поддерживающего WebRTC, с помощью SIPML5 можно совершать звонки или видеозвонки как по сети ТфОП, так и по сетям: SIP, LTE, IMS через шлюз webrtc2sip или шлюз webrtc2sip + сервер Asterisk 11. SIPML5 - это HTML5-приложение для VoIP, написанное на JavaScript, которое позволяет использовать браузер в качестве VoIP софтфона. SIPML5 - это клиентское приложение или SIP-клиент для браузера. Шлюз или серверное приложение webrtc2sip разработано Doubango Telecom - компанией Telco, которая специализируется на NGN технологиях [18]. Эта же компания разработала и клиентское приложение SIPML5, исходный код sipML5.js. [19].

Серверная программа webrtc2sip или шлюз, созданный на базе RTCWeb и SIP, состоит из четырех модулей: SIP Proxy, RTCWeb Breaker, Media Coder и Click-to-Call.

Реализация SIP-софтфона в web интерфейсе осуществляется на основе sipML5 API и webrtc2sip или на основе sipML5 API и коммуникационной платформы Asterisk (версия 11 и выше) с открытым исходным кодом в связке с webrtc2sip.

Методика разработки SIPml5-телефона в web интерфейсе:

1) установить бесплатное программное обеспечение webrtc2sip из SVN на платформу Ubuntu или установить webrtc2sip и Asterisk на платформу Ubuntu;

2) создать код клиентской части коммуникационного приложения на JS;

3) разработать интерфейс клиентского приложения с помощью гипертекстовой разметки HTML5 и CSS3.

Для разработки кода клиентской части SIP-телефона в web интерфейсе используется sipML5 API [20]. При разработке серверного приложения используется исходный код шлюза webrtc2sip [21, 22]. Серверное приложение webrtc2sip можно установить на платформу Ubuntu [23]. Установить webrtc2sip можно в соответствии с техническим руководством “Webrtc2sip - Smart SIP and Media Gateway for WebRTC endpoints” [24]. В случае использования коммуникационной платформы Asterisk и webrtc2sip, необходимо установить webrtc2sip между браузером и Asterisk [25]. Клиент, созданного веб-приложения (на основе HTML5+ CSS3+JavaScript), представляет собой программный SIPML5-телефон в web интерфейсе (softphone SIPML5 в браузере) без привязки к определенному VoIP провайдеру. Настройка

любого SIP HTML5 клиента для работы с Asterisk осуществляется в соответствии с руководством [26]. По функциональности web интерфейс SIPML5-софтфона или softphone SIPML5 в браузере приближен к клиентским приложениям, устанавливаемым на пользовательские устройства (ПК, смартфоны и т.д.) [27].

Суть работы веб-приложения sipML5 заключается в том, что звонки из браузера выполняются по протоколу сигнализации SIP, а передача медиа потоков между браузерами осуществляется с помощью WebRTC. При запуске клиентского приложения между клиентом и сервером устанавливается WebSocket-соединения. Для установления WebSocket-соединения клиент формирует особый запрос на основе HTTP (метод GET) с указанием того, что соединение должно быть обновлено до WebSocket-соединения. Сервер отвечает определённым образом, а код в строке состояния 101 означает, что WebSocket-соединения установлено, и осуществлен переход с HTTP на протокол WebSocket.

При вводе SIP-адреса в окне приложения sipML5, например sip:vova@sip2sip.info, и щелчке на кнопке "Позвонить" запускается клиентское приложение SIPml.js, которое описывает стек SIP/SDP на js. SIP – это сигнальный протокол, который работает совместно с протоколом описания параметров связи SDP. Сообщение протокола SDP передается в теле сообщения протокола SIP. Стек SIP/SDP обеспечивает обмен (передачу/прием) пакетов сигнального протокола SIP через установленное WebSocket-соединение между клиентом и приложением SIP Proxy. То есть, WebSocket-соединение используется в качестве транспорта для сигнального протокола SIP. Для транспортных протоколов TCP и UDP клиенты SIP используют порт 5060. В процессе установления сигнальной виртуальной SIP-сети между браузерами двух ПК, приложение sipML5 запрашивает у ПК разрешение на использование камер и микрофонов.

Получив разрешение, клиентские приложения с помощью WebRTC (RTCWeb) выполняют захват микрофонов и веб-камер, и создают каналы для передачи голосового и видео потоков. Захват микрофонов и веб-камер выполняет MediaStream (getUserMedia), а PeerConnection создает каналы для передачи голосового и видео потоков. WebRTC обеспечивает интеграцию средств голосовой и видеосвязи в режиме реального времени непосредственно в браузерах без установки дополнительных плагинов или расширений.

Голосовые и видео потоки между браузерами и шлюзом передаются по защищенным протоколам транспортного уровня DTLS-SRTP, а сессией управляет протокол SRTCP. DTLS - протокол датаграмм безопасности транспортного уровня, основанный на потоковом TLS, а SRTP - Безопасный Протокол Передачи Данных, с которым работает WebRTC. Эти протоколы применяются в связке DTLS-SRTP.

Между шлюзом и другими сетями с различными терминальными устройствами, которые не поддерживают WebRTC и SIPML5, передача пакетов сигнального протокола SIP осуществляется поверх UDP, TCP, TLS, которые поддерживаются всеми существующими сетями: SIP, LTE, IMS, а передача пакетов медиа потока осуществляется транспортными протоколами RTP и RTCP.

Стек протоколов, для p2p видеочата и SIP-телефон в web интерфейсе представлен на рисунке 6.

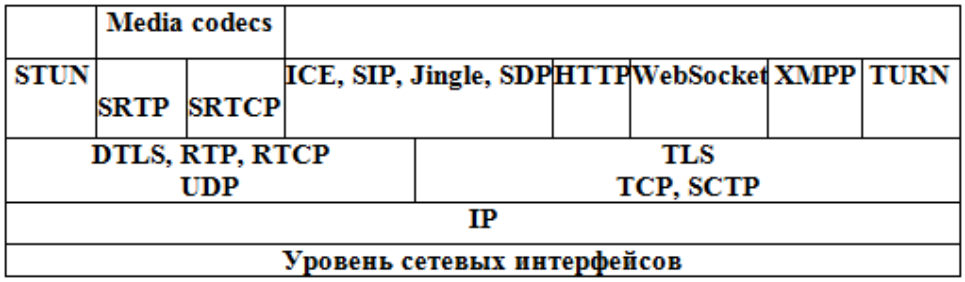

Рисунок 6 – Стек протоколов Web-коммуникационных сервисов в реальном времени

Следует отметить, что протоколы ICE, SIP, Jigle, SDP могут быть использованы поверх UDP, TCP и SCTP. ICE – это протокол установления интерактивного соединения с серверами STUN или TURN, работает совместно с протоколами STUN и TURN (ICE/STUN/TURN). Протокол Jigle можно использовать в качестве сигнального протокола для XMPP-сервера. SIP – это сигнальный протокол VoIP для SIP Proxy, который работает совместно с протоколом описания параметров связи SDP. В Web-приложениях реального времени допускается использование различных сигнальных протоколов, в том числе SIP или Jingle (Jingle/XMPP, Jingle RTP Sessions, Jingle ICE-UDP Transport Method, Jingle Raw UDP Transport Method), XMPP/Jabber. Протокол SDP используется для обмена описаниями характеристик пиров (медиа и других параметров) в WebRTC, SIP, Jingle/RTP, и передается в теле RTCSessionDescription или в сообщениях протоколов SIP и Jingle [28]. В традиционных SIP-сетях применяется один из стеков сигнального протокола: SIP/UDP/IP, SIP/TCP/IP или SIP/TLS/TCP/IP. В SIP-софтфоне с поддержкой WebRTC пакеты сигнального протокола SIP передаются через установленное WebSocket-соединение между клиентом и приложением SIP Proxy, т.е. применяется SIP/WebSocket.

На рисунке 6 протокол представительского уровня TLS условно отнесен к транспортному уровню, а протоколы для шифрования и дешифрования потока данных SRTP и SRTCP отнесены к прикладному уровню. Протоколы DTLS, RTP и RTCP работают поверх протокола UDP.

Результаты исследований и выводы. В результате исследований создана методика разработки веб-коммуникационных приложений с возможностями текстовой, аудио и видео связи между веб-браузерами. Предложенная методика обеспечивает разработку Web-коммуникационных сервисов в реальном времени: p2p видеочатов и SIPml5 софтфонов в web интерфейсе. Методика позволяет создавать собственные веб-приложения, на основе технологии HTML5 и WebRTC для голосовой и видеосвязи, а также для передачи файлов и обмена мгновенными сообщениями. Созданные Web-приложения реального времени можно интегрировать в сайты или другие веб-приложения и выполнять audio и video звонки непосредственно с этих сайтов или веб-приложений с помощью браузеров. WebRTC обеспечивает интеграцию веб - и телефонной связи и доступ к ним с помощью единого сетевого приложения - веб-браузера.

В основе разработанных методик лежат инновационные технологии HTML5 и WebRTC, которые позволяют интегрировать клиентские приложения p2p видеочат и VoIP софтфон в web-браузеры без привязки к конкретным пользовательским

устройствам (ПК, смартфонов, iPad, iPhone, IP-телефонов, мобильных телефонов и т.д.)

Необходимо отметить, что в настоящее ведутся работы по созданию пирингового видео по запросу P2P VOD (плеера пирингового ТВ) на основе WebRTC API без установки плагинов и расширений в браузере или клиентских приложений на ПК [28].

Список источников информации:

1. Бесплатные звонки через Интернет между устройствами с установленным Skype. [Электронный ресурс]. – Режим доступа: http://www.skype.com/ru/. 2. Miranda IM - Home of the Miranda IM client. Smaller, Faster, Easier. [Электронный ресурс]. – Режим доступа: http://www.miranda-im.org/. 3. WebRTC. [Электронный ресурс]. – Режим доступа: http://www.webrtc.org/. 4. HTML. [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/html/. 5. CSS Color Module Level 3. [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/css3-color/. 6. The WebSocket API. [Электронный ресурс]. – Режим доступа: http://dev.w3.org/html5/websockets/. 7. sipML5 - The world's first open source HTML5 SIP client. [Электронный ресурс]. – Режим доступа: http://sipml5.org/. 8. Webrtc2sip - Smart SIP and Media Gateway to connect WebRTC endpoints. [Электронный ресурс]. – Режим доступа: http://www.webrtc2sip.org/ 9. *Alan B. Johnston, Daniel C. Burnett*. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web. - St. Louis, USA: Digital Codex LLC, Smashwords Edition, 2013. – 247с. 10. *Ткаченко В.А.* P2P видеочат на базе WebRTC. [Электронный ресурс]. – Режим доступа: http://www.lessons-tva.info/articles/net/008.html. 11. WebRTC 1.0: Real-time Communication Between Browsers. [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/webrtc/. 12. Draft-moffitt-xmpp-over-websocket-00 - An XMPP Sub-protocol for WebSocket. [Электронный ресурс]. – Режим доступа: http://tools.ietf.org/html/draft-moffitt-xmpp-over-websocket-00. 13. XEP-0166: Jingle. [Электронный ресурс]. – Режим доступа: http://xmpp.org/extensions/xep-0166.html. 14. RFC 5245 - Interactive Connectivity Establishment. (ICE). [Электронный ресурс]. – Режим доступа: http://tools.ietf.org/html/rfc5245/ 15. OAuth на практике. [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/145988// 16. WebRTC/webRTC.io ·GitHub. [Электронный ресурс]. – Режим доступа: https://github.com/webRTC/webRTC.io. 17. webrtc.io-demo/site at master webRTC/webrtc.io-demo GitHub. [Электронный ресурс]. – Режим доступа: https://github.com/webRTC/webrtc.io-demo/tree/master/site. 18. Doubango Telecom. [Электронный ресурс]. – Режим доступа: http://doubango.org/. 19. This file is part of Open Source sipML5 solution. [Электронный ресурс]. – Режим доступа: http://sipml5.org/docgen/symbols/src/SIPml.js.html. 20. sipML5 Programmer's guide. [Электронный ресурс]. – Режим доступа: http://sipml5.org/docgen/index.html?svn=179. 21. SVN Checkout (исходный код). [Электронный ресурс]. – Режим доступа: http://webrtc2sip.googlecode.com/svn/trunk/webrtc2sip. 22. Git fork of the webrtc2sip doubango gateway. [Электронный ресурс]. – Режим доступа: https://github.com/stoiczek/webrtc2sip. 23. Installing webrtc2sip on Ubuntu 12.04 - linux.autostatic.com. [Электронный ресурс]. – Режим доступа: http://linux.autostatic.com/installing-webrtc2sip-on-ubuntu-1204. 24. Mamadou DIOP. Technical Guide. webrtc2sip - Smart SIP and Media Gateway for WebRTC endpoints.

[Электронный ресурс]. – Режим доступа: http://www.webrtc2sip.org/technical-guide-1.0.pdf. 25. База знаний -Настройка Asterisk -Связка WebRTC и Asterisk. [Электронный ресурс]. – Режим доступа: http://www.voxlink.ru/kb/asterisk-configuration/Asterisk-webrtc/. 26. Asterisk - The world's first HTML5 SIP client (WebRTC) - Google Project Hosting. [Электронный ресурс]. – Режим доступа: http://code.google.com/p/sipml5/wiki/Asterisk. 27. sipML5 demo [Электронный ресурс]. – Режим доступа: http://www.sipml5.org/call.htm. 28. SDP: Session Description Protocol. [Электронный ресурс]. – Режим доступа: http://tools.ietf.org/html/rfc4566
28. J. K. Nurminen, A. J. Meyn, E. Jalonen, Y. Raivio, and R. G. Marrero, “P2P media streaming with HTML5 and WebRTC,” in IEEE International Conference on Computer Communications. IEEE, 2013