

Felipe Silva Borges

WebRTC: Estudo e Análise do Projeto.

São José – SC

Março / 2013

Felipe Silva Borges

WebRTC: Estudo e Análise do Projeto.

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:

Prof. Ederson Torresini, M.SC.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Março / 2013

Monografia sob o título “*WebRTC: Estudo e Análise do Projeto.*”, defendida por Felipe Silva Borges e aprovada em 22 de março de 2013, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Ederson Torresini, M.SC.
Orientador

Prof. Deise Monquelate Arndt.
IFSC

Douglas Conrad
OpenS Tecnologia

As convicções são inimigas mais perigosas da verdade do que as mentiras.
Friedrich Nietzsche

Agradecimentos

Primeiramente Agradeço aos meus pais, por tudo que me proporcionaram, por não terem medido esforços me ajudarem a alcançar este objetivo.

Minha namorada Priscilla, pelo carinho e por me apoiar durante todo este período.

Agradeço também meu orientador Ederson, por sua ajuda, presteza e empolgação durante o decorrer deste trabalho.

Ao IFSC, por ter sido parte importante da minha vida, desde o ensino médio, e a todos os professores pois sem o conhecimento que transmitiram este trabalho não seria possível.

Resumo

WebRTC (*Real-Time Communications Web*) é um projeto recente, em fase de desenvolvimento cujo objetivo é o de prover comunicação em tempo real através da *Web* com alta qualidade de áudio e vídeo além de outras funções de transmissão de dados *peer-to-peer*, tudo isto utilizando apenas um navegador *Web* sem a necessidade de plugins adicionais ou softwares dedicados.

Este documento aborda os protocolos que trabalham em conjunto com WebRTC para que os serviços de comunicação em tempo real sejam possíveis. Também, trata das linguagens que permitiram o WebRTC se tornar realidade e ser relativamente simples de ser utilizado pelos desenvolvedores de aplicações *Web*.

O trabalho também faz um estudo sobre as novas funcionalidades trazidas pelo WebRTC, levando em conta as vantagens e desvantagens frente as atuais tecnologias e sempre que possível traça comparativos entre elas e verificando ainda quais seriam as possíveis utilizações desta tecnologia em um cenário comum de convergência telefonia-rede como, por exemplo, o IFSC câmpus São José.

Palavras-chave: WebRTC, Comunicação em tempo real, VoIP, Videoconferência.

Abstract

WebRTC (Real-Time Communications Web) is a recent project, being developed whose goal is to provide real-time communication through Web with high quality audio and video and other data transmission functions peer-to-peer, using just one browser Web without requiring additional plugins or dedicated software.

This document addresses the protocols that work in conjunction with WebRTC for services real-time communications are possible. Also, these languages which allowed the WebRTC become reality and be relatively simple to be used by developers Web of web applications.

The paper also makes a study of the new features brought by WebRTC, taking into account the advantages and disadvantages facing the current technologies and where possible draws comparisons between them and even checking what would be the possible uses of this technology on a common convergence scenario telephony-network such as the IFSC - São José.

Keywords: WebRTC, Real time communication, VoIP, Videoconferencing.

Sumário

Lista de Figuras

| | | |
|----------|---|-------|
| 1 | Introdução | p. 12 |
| 1.1 | Motivação | p. 14 |
| 1.2 | Objetivos | p. 14 |
| 1.2.1 | Objetivo Principal | p. 14 |
| 1.2.2 | Objetivos Específicos | p. 15 |
| 1.3 | Organização do texto | p. 15 |
| 2 | Comunicação em tempo Real | p. 16 |
| 2.1 | A evolução da comunicação através da Internet | p. 16 |
| 3 | WebRTC | p. 18 |
| 3.1 | Histórico | p. 18 |
| 3.1.1 | CU-RTC-WEB | p. 20 |
| 3.2 | Função do WebRTC | p. 21 |
| 3.3 | Aplicações existentes | p. 22 |
| 3.3.1 | Twinsee | p. 23 |
| 3.3.2 | Bistri | p. 23 |
| 3.3.3 | FrisB | p. 24 |
| 3.3.4 | Outras aplicações | p. 24 |
| 4 | Arquitetura WebRTC | p. 26 |

| | | |
|----------|--|-------|
| 4.1 | Web API | p. 27 |
| 4.1.1 | <i>MediaStream</i> | p. 27 |
| 4.1.2 | <i>PeerConnection</i> | p. 28 |
| 4.1.3 | <i>DataChannels</i> | p. 29 |
| 4.2 | WebRTC C++ API | p. 30 |
| 4.3 | Voice Engine | p. 30 |
| 4.3.1 | Codecs de Audio | p. 30 |
| 4.4 | Video Engine | p. 32 |
| 4.4.1 | Codecs de Vídeo | p. 32 |
| 4.5 | Sinalização | p. 33 |
| 4.5.1 | SIP | p. 33 |
| 4.5.2 | XMPP/Jingle | p. 35 |
| 4.5.3 | JSEP | p. 36 |
| 4.6 | Linguagens necessárias ao projeto. | p. 36 |
| 4.6.1 | HTML5 | p. 36 |
| 4.6.2 | Javascript | p. 37 |
| 4.6.3 | Segurança | p. 37 |
| 4.6.4 | Navegadores | p. 38 |
| 5 | Cenários e Testes | p. 41 |
| 5.1 | Cenários | p. 41 |
| 5.1.1 | Comunicação entre navegadores num mesmo WebServer | p. 42 |
| 5.1.2 | Comunicação entre navegadores com WebServers Distintos | p. 42 |
| 5.1.3 | Integração com outras tecnologias | p. 43 |
| 5.1.4 | Integração com SIP | p. 43 |
| 5.1.5 | Integração com Telefonia Fixa | p. 50 |

6 Conclusões

p. 52

Referências Bibliográficas

p. 55

Lista de Figuras

| | | |
|-----|--|-------|
| 1.1 | Comparativo minutos VoIP x TDM | p. 13 |
| 1.2 | Crescimento do Skype diante TDM e VoIP | p. 13 |
| 3.1 | Relação entre <i>Browser</i> e <i>WebServer</i> | p. 20 |
| 3.2 | Comunicação entre Navegadores | p. 22 |
| 3.3 | Desenvolvedores do WebRTC utilizando o Twinsee | p. 23 |
| 3.4 | FrisB | p. 24 |
| 4.1 | Arquitetura do WebRTC | p. 26 |
| 4.2 | MediaStream | p. 27 |
| 4.3 | Exemplo MediaStream | p. 28 |
| 4.4 | Exemplo PeerConnection | p. 29 |
| 4.5 | Gráfico Comparativo de Codecs | p. 32 |
| 4.6 | MediaStream | p. 35 |
| 4.7 | Autorização para utilização de recursos | p. 38 |
| 4.8 | Suporte nos navegadores de Desktop | p. 39 |
| 5.1 | Relação entre Browser e WebServer | p. 41 |
| 5.2 | Comunicação entre dois Navegadores usando WebRTC através de um <i>Web-Server</i> | p. 42 |
| 5.3 | Apprtc em funcionamento | p. 43 |
| 5.4 | Troca de mensagens entre Navegadores e <i>WebServer</i> | p. 44 |
| 5.5 | Dois navegadores usando servidores distintos | p. 45 |
| 5.6 | Troca de mensagens entre Navegadores e <i>WebServer</i> | p. 45 |
| 5.7 | WebRTC com SIP | p. 46 |

| | | |
|------|--|-------|
| 5.8 | Configuração de uma conta com o SIPML5 | p. 47 |
| 5.9 | Diagrama WebRTC2SIP | p. 48 |
| 5.10 | Cenário de testes | p. 48 |
| 5.11 | Troca de mensagens WebRTC operando em conjunto com SIP | p. 49 |
| 5.12 | Troca de mensagens WebRTC2SIP e Asterisk | p. 50 |
| 5.13 | WebRTC com telefonia fixa | p. 50 |

1 *Introdução*

Ao longo dos séculos o homem sempre sentiu necessidade de se expressar dentro de um círculo social. Isso fez com que ele sempre buscasse evolução na forma de transmitir suas ideias e necessidades ao próximo e pensando nisso surgiram e aprimoraram-se os chamados meios de comunicação.

Com o surgimento da Internet, começou uma revolução na forma de comunicação que havia até então. Pode-se dizer que a Internet foi desde o princípio considerada como um caminho para a evolução dos meios de comunicação.

Surgiram várias ideias de aplicações que utilizassem comunicação através de áudio e vídeo, porém, no princípio essas ideias não eram muito viáveis. Em sua maioria, os sistemas eram dependentes de *hardware* e rede específicos com um alto custo de implantação, principalmente quando falamos em relação a videoconferência, além do fato de que existiam grandes problemas de interoperabilidade devido a soluções proprietárias de diversos fornecedores.

Com a evolução da internet, aumento da banda disponível e melhoria no *hardware* dos computadores, foi possível fazer serviços cada vez melhores e mais acessíveis, até chegarmos ao ponto em que estamos hoje, em que existe uma grande variedade de serviços e aplicações voltadas tanto para telefonia VoIP quanto serviços de videoconferência.

Podemos notar em pesquisas como a que foi realizada pelo site Telegeography (2013), exemplificada pelo gráfico da Figura 1.1 que a quantidade de minutos realizados na rede TDM vem estagnando, principalmente nos últimos 4 anos enquanto as chamadas baseadas em VoIP continuam crescendo.

Esta mudança é ainda mais impressionante se for observado pela visão das OTTs (*over-the-top*), empresas que tem como foco a prestação de serviços de comunicação através da Internet. O principal exemplo desses casos é o Skype, que permite a comunicação entre seus usuários com áudio e vídeo.

Na Figura 1.2, também retirada da pesquisa do site *TeleGeography* esta situação fica bem

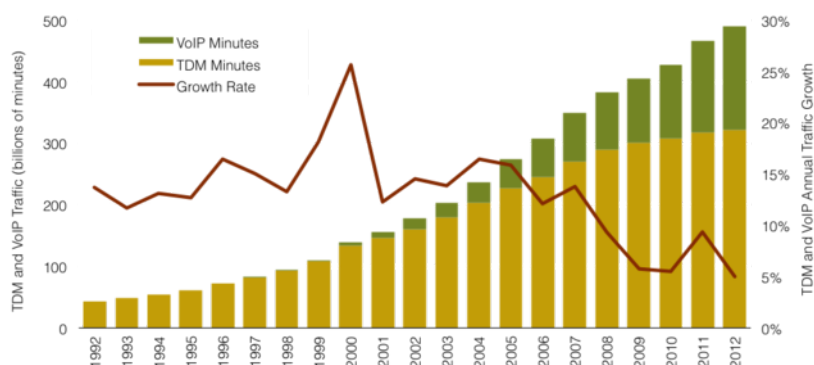


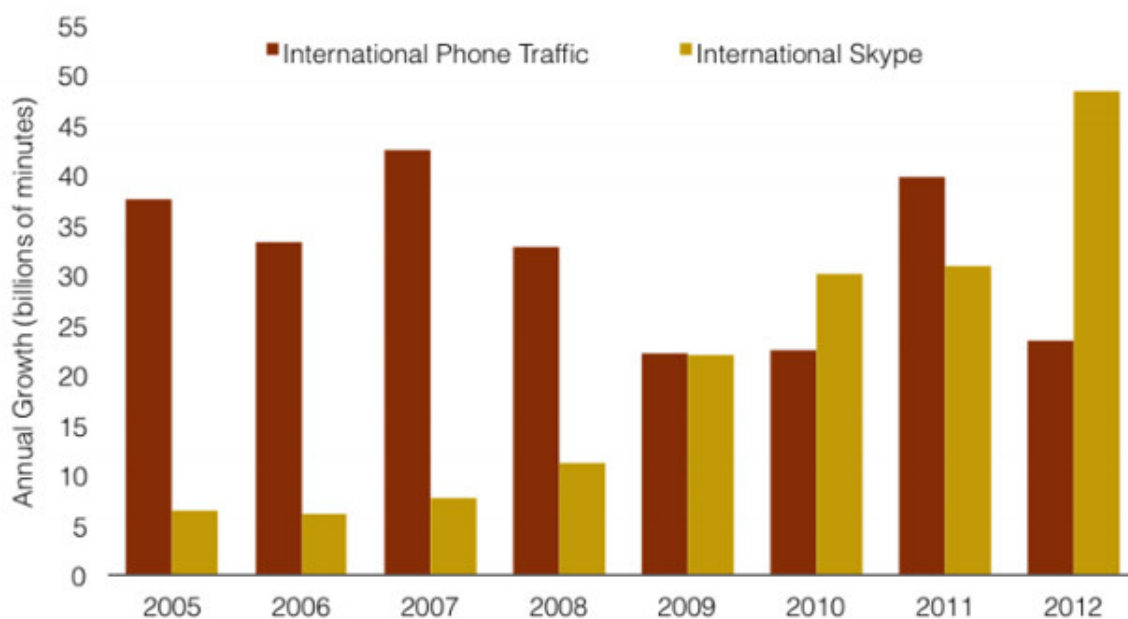
Figura 1.1: Comparativo minutos VoIP x TDM

Fonte: Telegeography, 2013

evidente. É feito um comparativo entre o crescimento do Skype¹ frente ao mercado TDM e VoIP prestado por operadoras.

FIGURE 7

Increase in International Phone and Skype Traffic, 2005-2012



Notes: ILD traffic reflects TDM and VoIP. Skype traffic growth reflects Skype-to-Skype traffic, including video calls. Skype calls to the PSTN are excluded.

Source: TeleGeography

© 2013 PriMetrica, Inc.

Figura 1.2: Crescimento do Skype diante TDM e VoIP

Fonte: Telegeography, 2013

¹ <http://www.skype.com/>

É possível notar um crescimento constante do Skype enquanto os outros serviços tem passado por uma clara variação do crescimento nos últimos anos.

Vale destacar também o surgimento de vários serviços OTTs (*Over-the-top*) que vem ganhando espaço no mercado, principalmente graças a sua facilidade de uso nas plataformas móveis como por exemplo o *WhatsApp*¹ e o *Viber*²

Entretando ainda há muito para ser feito, as aplicações que temos hoje em dia são dependentes de clientes dedicados ou *plugins*, e é neste ponto que está focado o projeto WebRTC, alvo de estudo deste trabalho, que pretende permitir comunicação e colaboração em tempo real, através de áudio, vídeo e troca de arquivos utilizando apenas um navegador de internet sem nenhum *software* adicional.

É nesse ponto que o projeto WebRTC pode realizar grandes mudanças, já que propõe soluções simples para problemas que temos hoje quando se fala em comunicação em tempo real através da web, além de ser impulsionado por grandes empresas da áreas de telecomunicações e *Internet*.

1.1 Motivação

O WebRTC vem para agregar novas funcionalidades a essa realidade, permitindo a criação de aplicações para estas finalidades de uma forma simples e até então nunca vista, fazendo com que os navegadores *web* ganhem outros papéis até então impensados para eles. Porém cabe reforçar que ainda é um projeto em fase desenvolvimento.

1.2 Objetivos

1.2.1 Objetivo Principal

Este trabalho tem por objetivo estudar e analisar o Projeto WebRTC, com o intuito de verificar as possibilidades apresentadas por esta tecnologia, bem como suas vantagens e desvantagens em relação a outras tecnologias presentes e difundidas.

¹<http://www.whatsapp.com/>

²<http://www.viber.com/>

1.2.2 Objetivos Específicos

- Realizar testes para analisar o funcionamento da tecnologia em um cenário como o IFSC câmpus São José.
- Verificar quais os benefícios e possibilidades o WebRTC poderia promover se fosse integrado a outras ferramentas que já são utilizadas no câmpus.

1.3 Organização do texto

O texto deste trabalho está organizado na forma de capítulos, no capítulo 2 é feita uma revisão sobre alguns conceitos básicos, de comunicação em tempo real bem como um breve histórico sobre o processo de evolução da comunicação e as tecnologias que nos trouxeram até aqui. Também será falado sobre algumas tecnologias de comunicação em tempo real que temos hoje em dia, sejam elas baseadas em *web* ou não.

Já no capítulo 3 define a função do projeto WebRTC (*Real time communication Web*) em um sistema de comunicação em tempo real para web em seguida, o 4 trata sobre o funcionamento do projeto, levando em conta sua arquitetura e a forma que executa as funções atribuídas à ele. Também são abordados os protocolos que trabalham em conjunto com o WebRTC.

Finalmente, no capítulo 5 são exibidos alguns dos possíveis cenários de implantação da tecnologia WebRTC, e os testes realizados com diversas aplicações.

Este trabalho está basicamente baseado nos *drafts* do IETF e W3C, por se tratar de um assunto novo, são poucas as fontes de informação à respeito dele.

2 *Comunicação em tempo Real*

A comunicação se desenvolveu junto com a evolução humana e é considerada, segundo Rodrigues (2007), o viés mais importante deste processo. Ela não tem uma data de início, mas pode-se dizer que ela sempre existiu, desde os primórdios. Não existe humanos sem comunicação, a história da comunicação funde-se com a história humana. Quando se comunica o homem se descobre, descobre o mundo, cria códigos e estabelece hierarquias, ou seja, faz a sociedade em que vive evoluir junto com as suas relações interpessoais.

Neste século os rumos da comunicação mudaram de forma que o homem não imaginava e a evolução da comunicação e dos seus meios começou a se acelerar drasticamente. Fazendo a comparação por exemplo entre a evolução do rádio e da *internet*, em que o rádio levou 20 anos para surgir e se tornar um meio viável de comunicação, enquanto a *internet* levou apenas 5 anos para fazer o mesmo.

2.1 A evolução da comunicação através da Internet

Na década de 60 houve a criação da ARPA (Advanced Research Project Agency, ou Agência de Pesquisas em Projetos Avançados), e em muito se deve a necessidade da troca de informações de forma segura durante a Guerra Fria, sendo que em 1969 a ARPANET obteve sucesso na transmissão de dados.

A década de 1970 foi responsável pela criação dos conceitos básicos de rede, como por exemplo a criação da “*interneting*” em 1971, que consistia em conectar três redes diferentes no que se tornaria mais tarde a Internet como conhecemos hoje. Ainda na mesma década o engenheiro Ray Tomlinson combinou um aplicativo de troca de mensagens chamado SNDMSG com um protocolo CYPNET possibilitando a transmissão em rede e posteriormente foi utilizado o termo @ para diferenciar o nome do usuário do seu servidor. Porém entre todas as mudanças que aconteceram neste período a que mais importou para o amadurecimento da Internet como conhecemos hoje foi a criação do TCP/IP (*Transmission Control Protocol / Internet Protocol*)

que é um protocolo padrão para a transmissão de dados até hoje.

Com a chegada dos anos 80 a rede já estava bastante desenvolvida e os programadores expandiam as fronteiras da rede, criando novas funções para a rede que estava se popularizando. Começam a engatinhar nas indústrias então a ideia do computador pessoal, com o lançamento por exemplo do IBM PC em 1981. A partir deste lançamentos este estilo foi copiado por outras empresas ao longo dos anos e com o surgimento da Microsoft e a sua concorrência com a Apple ditaram o ritmo do crescimento dessa nova geração de computadores.

No final da década de 1980 surge então o IRC inicialmente criado para a troca de mensagens durante a Guerra do Golfo e que serviu de base para diversos mensageiros instantâneos como conhecemos hoje. Então na mudança para a década de 1990 a Internet já estava consolidada como uma das grandes forças da tecnologia, porém o futuro guardava ainda mais surpresas e novidades para este serviço de rede como a criação do *World Wide Web* em 1989 como um projeto de hipertextos para dinamizar a passagem de um texto a outro de forma mais rápida e dinâmica neste sistema que entraria em funcionamento na década seguinte com a ajuda de Robert Cailliau que reformulou o projeto e formalizou o mesmo despertando o interesse de universidades que utilizavam outras redes.

Além da criação do *World Wide Web*, Tim Bernes-Lee ajudou a desenvolver também algumas bases desse sistema de navegação por cliques que são o HTTP e HTML. Com as invenções de Bernes Lee e as melhorias no códigos e protocolos chegamos a Internet como conhecemos hoje. A Internet agora era coisa séria e foram criadas formas de integrar serviços já existentes, como a NSFnet. Sendo que a comercialização e unificação da Internet foram encaminhados em 1993, porém em dois anos a NSFnet deixou de ser “dona” das redes e todo o tráfego passou a ser público. Começaram então a surgir o domínio de *sites* comerciais e a Internet então começou a crescer desenfreadamente a partir da conhecida “bolha da Internet”.

A Internet permitiu uma grande evolução nos serviços de comunicação. No início conforme falado anteriormente, com aplicações de chat como IRC e serviço de e-mail, com o passar do tempo surgiram novos serviços de mensagem instantânea e voz, chegando até as videochamadas, podemos dizer que estes dois últimos são serviços que vem crescendo cada vez mais, segundo estimativa da Cisco (2012), o tráfego de vídeo na internet representará 56% do total trafegado, desconsiderando vídeos compartilhados via P2P.

3 *WebRTC*

3.1 Histórico

WebRTC é um projeto de código aberto que foi criado com o intuito de permitir aplicações de comunicações em tempo real com vídeo e áudio utilizando apenas navegadores *Web*, sem a necessidade de *plugins* e com grande qualidade de voz e imagem. Pela primeira vez, será possível que dois ou mais navegadores interajam diretamente com o intuito de transmitir em uma conexão *peer-to-peer* áudio, vídeo e dados.

Segundo Uberti (2012a)¹, durante sua apresentação no evento Google IO, a ideia para o projeto surgiu quando uma equipe de engenheiros do Google responsável pelo desenvolvimento dos *hangouts* (Serviço que permite comunicação de vários usuários com áudio e vídeo) do Google+ notou que havia um grande problema para o desenvolvimento de aplicações para *Web*: o uso de *plugins* e os problemas que eles trazem como por exemplo, instalações adicionais, versões e questões de segurança. Neste momento, surgiu a ideia da criação de uma plataforma de código aberto que não necessitasse mais da utilização de *plugins*.

A partir daí o Google deu início a execução deste projeto, a princípio sem incluir outras empresas. Primeiramente efetuou em fevereiro de 2010 a compra da empresa On2, detentora dos direitos do *codec* VP8 (Este assunto será tratado na seção 4.4.1). O próximo passo foi a aquisição da empresa GIPS (*Gobla IP Solutions* em maio de 2010, esta corporação desenvolveu uma série de soluções de processamento de áudio e vídeo para redes IP York (2010).

Neste momento o Google já possuía as ferramentas básicas para o lançamento de sua proposta e deu início a fase padronização, realizando consultas a diversas empresa dos setores de telecomunicações, *internet* e desenvolvedores de *softwares* com o intermédio do W3C (*World Wide Web Consortium*).

Empresas como:

¹http://www.youtube.com/watch?feature=player_embedded&v=E8C8ouiXHHk

- Ericsson - Fabricante de produtos e soluções na área de telecomunicações;
- Mozilla - Desenvolvedora do navegador Firefox;
- Opera - Desenvolvedora do navegador Opera;
- Cisco - Fabricante de produtos e soluções na área de telecomunicações;
- AT&T - Operadora de telecomunicações;

A resposta obtida foi muito positiva, dando início desta forma ao processo de padronização da proposta. Foram criados grupos de discussão no IETF (*Internet Engineering Task Force*) e W3C (*World Wide Web Consortium*) com objetivo de criar um padrão que permita que qualquer navegador se comporte nativamente como uma plataforma de comunicação em tempo real através da *web*, que seja multiplataforma e amplamente aceito pela indústria.

Uma das grandes vantagens do WebRTC é o fato de ser um projeto de código aberto sob uma licença BSD¹ (*Berkeley Software Distribution*), que está buscando utilizar em sua grande maioria, *codecs* livres de *royalties* permitindo que os desenvolvedores possam criar as mais diversas aplicações para comunicação em tempo real de uma maneira prática.

O desenvolvimento das aplicações é feito a partir de APIs (*Application Programming Interface*) Javascript, sem que o desenvolvedor tenha que necessariamente ter amplos conhecimentos sobre *codecs* de áudio e vídeo, cancelamento de eco, redução de ruídos, enfim, diversas técnicas que neste caso são tratadas pelo WebRTC, deixando o desenvolvedor focado apenas na aplicação, as funções citadas passam a ser responsabilidade do navegador *web* desde que este tenha suporte ao projeto WebRTC.

A Figura 3.1 mostra como é a relação entre um navegador contendo WebRTC e um Servidor *Web*, passando pelas APIs de programação Javascript.

O quadrado mais claro da Figura 3.1 (*browser RTC function*) será alvo de estudo na seção 4.

O Projeto WebRTC vem sendo definido por uma série de recomendações e *drafts* dos órgãos IETF (*Internet Engineering Task Force*) e W3C. O W3C é responsável pela definição da API necessária para que as aplicações em Javascript interajam com o navegador. Já o IETF está cuidando do desenvolvimento dos protocolos que serão utilizados na comunicação de um navegador com outro ou com outro *endpoint*² qualquer em casos de interconexão com outras

¹Licença de código aberto

²Ponto para onde a chamada de vídeo ou áudio será enviada

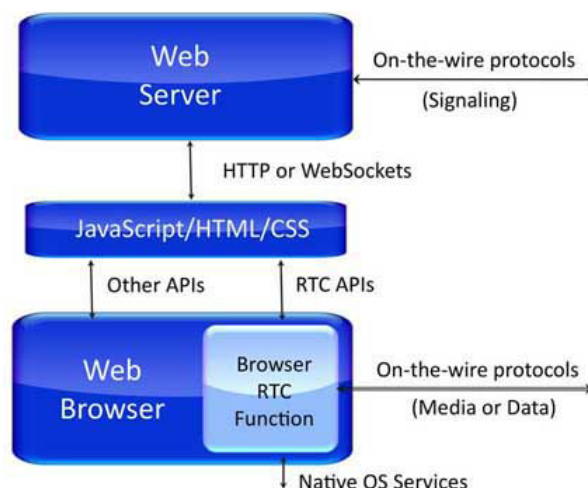


Figure 1.2 The Browser Model

Figura 3.1: Relação entre *Browser* e *WebServer*

Fonte: Johnston , 2012

tecnologias. Vale lembrar que por se tratar de um projeto recente, a grande maioria dos documentos criados ainda não são definitivos, não há por exemplo, nenhuma RFC(*Request For Comments*) pronta diretamente relacionada ao WebRTC, apenas *drafts*.

Segundo Alvestrand (2013) , o objetivo desta colaboração entre a especificação do protocolo e da API, é permitir que todas as características e opções do protocolo sejam dispostas de maneira clara. Segundo o mesmo documento, a obrigatoriedade de implementação de um conjunto mínimo de funções garante o desenvolvimento do mercado que utilizará WebRTC, ou seja, se todas as partes estiverem em conformidade com a especificação, elas poderão se comunicar independente do fornecedor da solução.

Porém a especificação não proíbe funcionalidades adicionais além das especificadas inicialmente, sendo o desenvolvedor livre para adicionar novas funcionalidades para uma determinada aplicação ou cenário específico.

3.1.1 CU-RTC-WEB

Uma das empresas inicialmente consultadas para ajudar na determinação dos padrões do WebRTC, a Microsoft discorda das outras empresas em determinados pontos e, por isso, sugeriu uma outra versão do WebRTC denominada CU-RTC-WEB (*Customizable, Ubiquitous Real Time Communication over the Web*) que segundo Roettgers (2012) tem como maiores diferenças:

- Adoção de múltiplos *codecs* de vídeo - A Microsoft defende que mais de um *codec* de

vídeo, enquanto as outras empresas defendem que para facilitar a padronização deve ser utilizado apenas um (VP8 ou H.264, ainda não há uma definição como veremos na seção 4.4.1).

- Permitir que o desenvolvedor tenha mais acesso as camadas inferiores do WebRTC, desta forma, dando uma maior flexibilidade. O contraponto desta proposta é fato de que o desenvolvimento de aplicações seria mais complexo, já que os desenvolvedores teriam que lidar com mais variáveis que na versão previamente proposta.

3.2 Função do WebRTC

Conforme falado anteriormente, o objetivo do WebRTC é permitir a utilização de serviços de comunicação em tempo real através da *web*, entretanto, o WebRTC não fornece uma solução completa para um sistema que visa permitir este tipo de comunicação, sendo que o WebRTC terá um papel fundamental, o qual será abordado a seguir, portanto é preciso ressaltar que ele não é a única ferramenta que será utilizada para permitir a comunicação entre os usuários.

A função base do WebRTC é servir como um *media engine*¹, interagindo com as aplicações *web* através das APIs que estão sendo padronizadas, também interage com o sistema operacional através do browser para que seja possível utilizar os recursos do usuário, como microfone e câmera.

A Figura 3.2 ilustra esta divisão das funções entre WebRTC/Browser e aplicações *Web*. É importante ressaltar nesta figura, um dos grandes diferenciais do WebRTC, que a mídia fica entre os dois navegadores.

Reforçando o que foi dito anteriormente, o Alvestrand (2013) define que, a ideia não é permitir que um navegador tenha todas as funções necessárias para se comportar como um telefone ou uma unidade de vídeoconferência, mas sim garantir que o navegador tenha as funções necessárias para que uma aplicação *Web* implemente estas funções.

Com isso, podemos dizer que são funções de um navegador com suporte a WebRTC, a codificação/decodificação das mídias que serão utilizadas, bem como funções de processamento de mídia como cancelamento de eco entre outras.

Outras funções são de responsabilidade da aplicação, que podem ser adotadas de acordo com a finalidade ou preferência do desenvolvedor, como por exemplo a sinalização. O projeto do WebRTC não define qual sinalização deve ser utilizada, desta forma permite uma grande

¹ Responsável por realizar o tratamento da mídia.

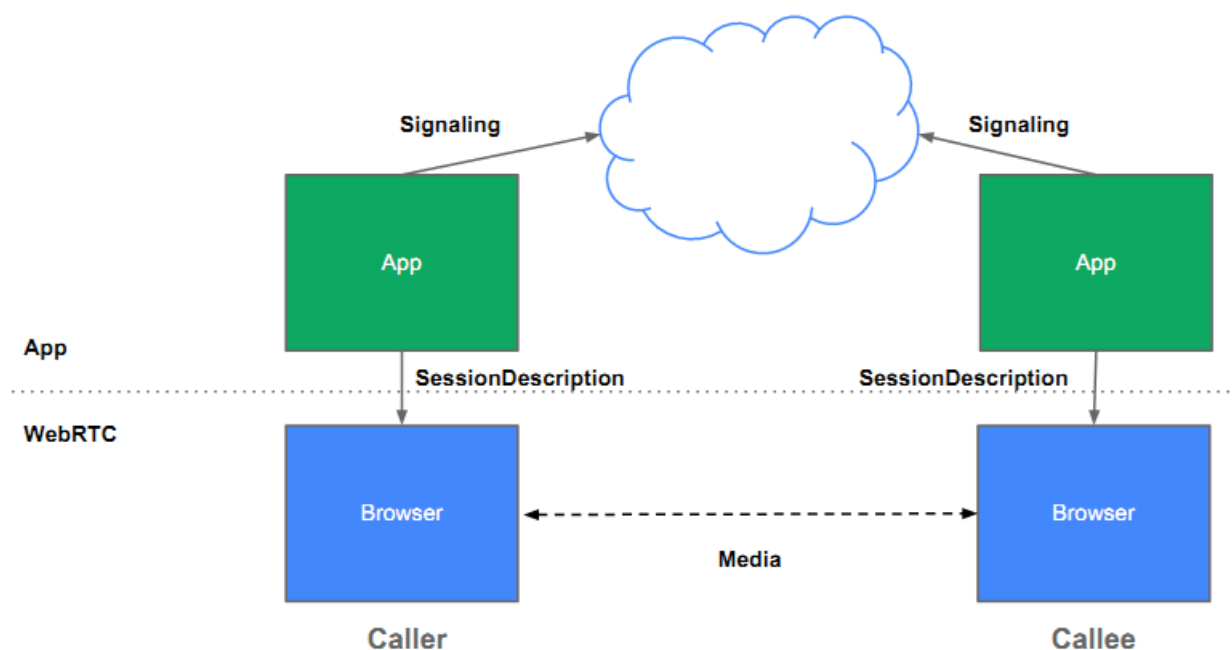


Figura 3.2: Comunicação entre Navegadores

Fonte: Uberti, 2012

flexibilidade já que podem ser utilizadas sinalizações amplamente difundidas como SIP (ROSENBERG, 2002), XMPP (SAINT-ANDRE, 2004), H.323 (ITU-T, 2009), ou sinalizações proprietárias e até mesmo propostas como JSEP (UBERTI, 2012b) e ROAP (JENNINGS, 2012) que estão sendo pensadas justamente para trabalharem com o WebRTC. Este assunto será abordado com maior profundidade na seção 4.5.

3.3 Aplicações existentes

Já existe uma boa quantidade de aplicações que utilizam WebRTC como tecnologia base. Boa parte ainda está em fase de testes ou são aplicações de demonstração, porém, apesar do projeto ainda nem estar completamente padronizado, já existem algumas aplicações comerciais utilizando o WebRTC como solução.

Olhando para as aplicações já existentes, é possível ter uma ideia das funcionalidades que o WebRTC permite implementar. Seguem algumas delas:

3.3.1 Twinsee

*Twinsee*¹ é um serviço de mensagens instantâneas e videochamadas de alta qualidade, onde o usuário tem uma lista com seus contatos organizados na forma de círculos.

Para as videochamadas, é utilizado o WebRTC com mídia *peer-to-peer*, enquanto que para o serviço de mensagens instantânea é utilizado o protocolo XMPP.

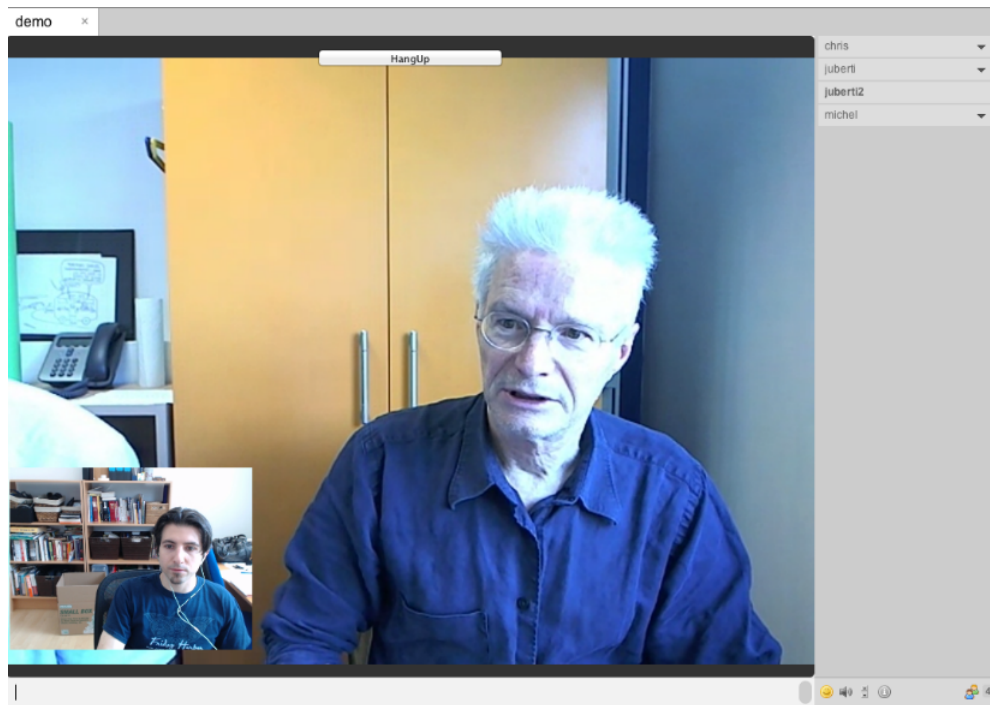


Figura 3.3: Desenvolvedores do WebRTC utilizando o Twinsee

Olhando a Figura 3.3 é possível ver o Twinsee em funcionamento, com duas pessoas conversando através de uma videochamada com as conversas via chat ao lado.

3.3.2 Bistri

O Bistri² tem uma função semelhante do *Twinsee*, ou seja videochamadas e mensagens instantâneas, porém o Bistri é um ferramenta que pretende ser uma rede social de comunicação em tempo real. Nele o usuário pode agrupar todas as suas contas de outros serviços e são adicionados seus respectivos contatos, a ideia é que um usuário possa por exemplo, falar com seus contatos do Skype, Facebook, Gmail, enfim, quantas contas desejar com a vantagem de ter uma única plataforma para isso com suporte a videochamadas e outros recursos.

¹<https://twinsee.net/>

²<https://bistri.com/>

3.3.3 FrisB

O FrisB¹ é basicamente um serviço de *callback*², porém extremamente interessante, ele consiste apenas de um portal *Web* com espaço para discagem, como mostra a Figura 3.4. O usuário discar para um número em qualquer lugar do mundo, o serviço faz a discagem e depois de determinado tempo desconecta a chamada evitando desta forma a tarifação.

A pessoa que recebeu a chamada liga de volta para o número de onde recebeu a chamada e então a chamada é encaminhada para o navegador da pessoa que fez a primeira ligação.

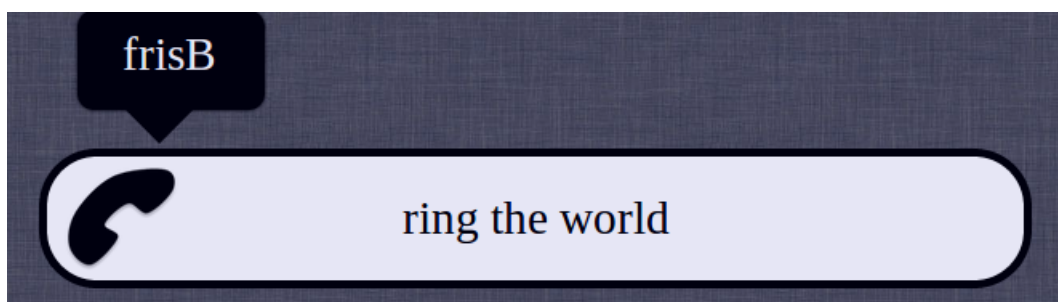


Figura 3.4: FrisB

A vantagem desta aplicação é o custo, por exemplo, quando alguém da Europa liga pra um telefone no Brasil, o número de origem consta como um telefone brasileiro com DDD 21, quando a pessoa retorna para este número a chamada é completada e será cobrado uma ligação local ou longa distância nacional, dependendo do ponto onde ela estiver, mas não pagará uma ligação longa distância internacional por exemplo. Atualmente o FrisB é a única aplicação que faz interligação entre WebRTC e STFC sem cobrar do usuário por isso.

Nos testes realizados com essa ferramenta, percebemos um pequeno atraso, mas que não prejudicou a conversação e uma excelente qualidade de áudio.

3.3.4 Outras aplicações

Há uma série de outras aplicações que estão utilizando WebRTC, muitas soluções já existentes estão migrando suas plataformas como é o caso por exemplo da *meetings.io*³ que fornece salas de videoconferência gratuitamente, e que hoje em dia utilizam o plugin do *Flash player* para funcionar.

Outras aplicações que utilizam WebRTC serão mostradas no capítulo 5 em conjunto com

¹<http://www.frisb.com/>

²Quando é originada uma segunda chamada em função da primeira chamada recebida.

³<https://meetings.io/>

os cenários estudados e testes realizados.

4 Arquitetura WebRTC

A arquitetura do projeto WebRTC se divide basicamente em duas diferentes camadas. A primeira camada, denominada Web API onde serão desenvolvidas as aplicações que utilizarão o WebRTC, e a segunda camada que é o WebRTC propriamente dito contendo a API de desenvolvimento de *browsers*¹ e os demais *stacks*² presentes no projeto (áudio, vídeo, transporte, etc.) (GOOGLE, 2012a). A Figura 4.1 demonstra a abstração da arquitetura do WebRTC, com suas camadas e blocos.

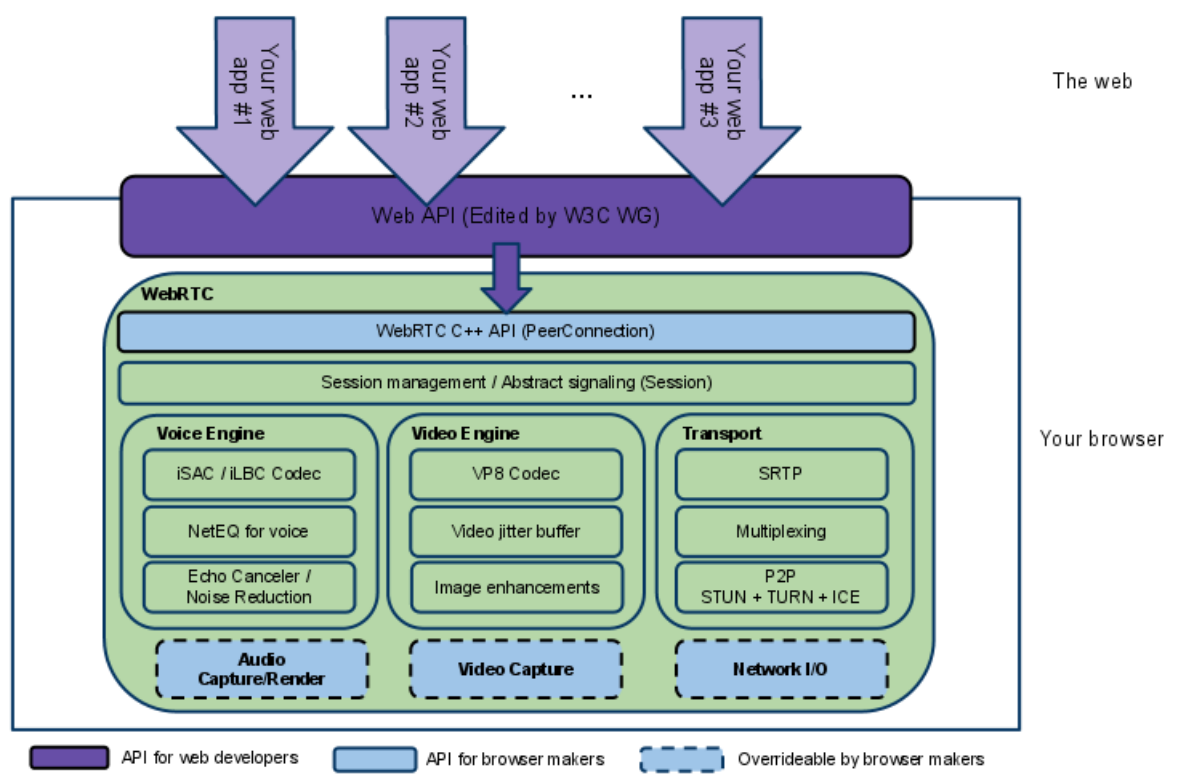


Figura 4.1: Arquitetura do WebRTC

Fonte: *webrtc.org*, 2012

Os principais itens da arquitetura serão aprofundados na sequência deste capítulo.

¹Navegador Web

²Estrutura de dados em forma de pilha.

4.1 Web API

Esta é a API para desenvolvimento das aplicações de comunicação em tempo real via *web*. Basicamente podemos dividi-la em 3 diferentes sub-APIs, que poderão ser utilizadas de diferentes maneiras para a obtenção de diversas funcionalidades pelos desenvolvedores de aplicações *web* (GOOGLE, 2012b), são elas:

- *MediaStream*
- *PeerConnection*
- *DataChannels*

4.1.1 *MediaStream*

MediaStream é a função responsável por fornecer uma fonte de mídia para o WebRTC. É esta API responsável por solicitar acesso ao microfone e a câmera do usuário e capturar o fluxo de dados fornecidos por estes equipamentos. Um *MediaStream* pode conter um ou mais *MediaStreamTracks*, por exemplo, em uma chamada áudio e vídeo de um participante formam um *MediaStream*, sendo que o áudio será um *MediaStreamTrack* e o vídeo outro, a Figura 4.2 ilustra bem o conteúdo de um *MediaStream* Padrão (O'CALLAHAN, 2012)

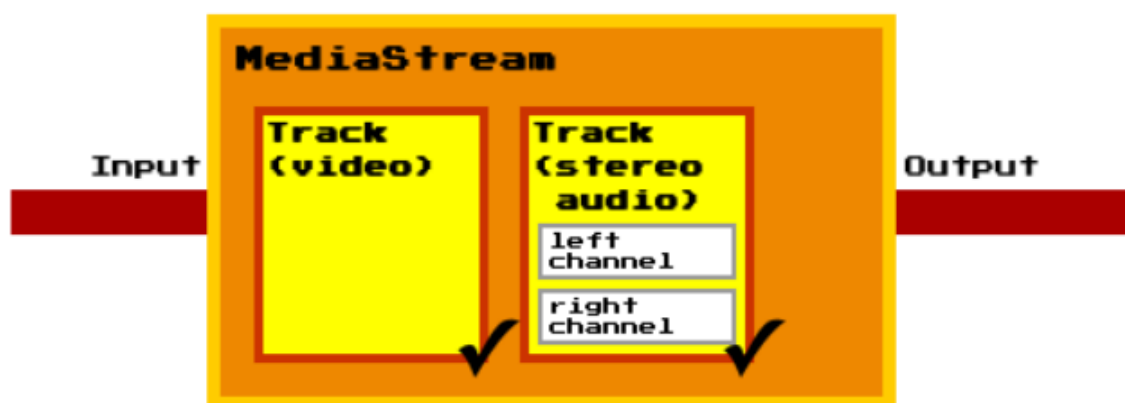


Figura 4.2: *MediaStream*

Fonte: Uberti, 2012

Não há um limite teórico para a quantidade de *MediaStreamTracks* dentro de um *MediaStream*, ou seja, um usuário pode estar usando duas ou mais câmeras, várias fontes de áudio, e mesmo assim todas essas informações podem ser transmitidas, desde que a aplicação web permita isso, este é outro fator que amplia as possibilidades a serem exploradas com o WebRTC.

Outra função presente nos *MediaStream* é a capacidade de permitir ao usuário que escolha a resolução do vídeo que está sendo exibido.

Por razões de segurança, sempre que um *MediaStream* é solicitado a um usuário, este deve aceitar compartilhar seu áudio/vídeo através de um *prompt*.

O código da Figura 4.3 foi retirado de (UBERTI, 2012c), e implementa uma aplicação¹. de teste que tem como objetivo apenas demonstrar como é simples realizar a aquisição do vídeo de um usuário através da função *getUserMedia*.

```
1 <script type='text/javascript'>
2 navigator.webkitGetUserMedia({video:true}, onGotStream,
3 onFailedStream);
4 onGotStream = function(stream) {
5 var url = webkitURL.createObjectURL(stream);
6 video.src = url;
7 }
8 </script>
9 <video id="video" autoplay="autoplay" />
```

Figura 4.3: Exemplo *MediaStream*

Fonte: Uberti, 2012

4.1.2 *PeerConnection*

Segundo Dutton (2013) esta API é responsável por prover aos usuário uma conexão estável e eficiente. É ela quem realiza estabelecimento de uma conexão direta entre dois navegadores, e faz o controle do *codec* que será utilizado, criptografia e gerenciamento de banda.

O Código mostrado na figura 4.4, implementa as funções de enviar e receber uma conexão através da API *PeerConnection*.

Nas linhas 2 e 3 da Figura 4.4, temos os comandos referentes a criação de dois *PeerConnection*, sendo um pra envio e um pra recebimento, em seguida, na linha 4, a mídia local que está sendo gerada neste caso pela webcam, é inserida dentro do *PeerConnection* de envio. Depois disso temos a negociação do SDP, nas linhas de 7 a 11, e finalmente, a conexão é realizada na linha de número 12.

Se for analisado todo o procedimento por trás, pode-se dizer, que é um código bastante simples.

¹<http://webrtc.googlecode.com/svn/trunk/samples/js/demos/html/gum1.html>

```
1 <script type='text/javascript'>
2 pc1 = new webkitPeerConnection00(null, onIceCandidate1);
3 pc2 = new webkitPeerConnection00(null, onIceCandidate2);
4 pc2.onaddstream = onRemoteStream;
5 pc1.addStream(localStream);
6 var offer = pc1.createOffer(null);
7 pc1.setLocalDescription(pc1.SDP_OFFER, offer);
8 pc2.setRemoteDescription(pc2.SDP_OFFER, offer);
9 var answer = pc2.createAnswer(offer.toSdp(), null);
10 pc2.setLocalDescription(pc2.SDP_ANSWER, answer);
11 pc1.setRemoteDescription(pc1.SDP_ANSWER, answer);
12 pc1.startIce(); pc2.startIce();
13 </script>
```

Figura 4.4: Exemplo PeerConnection

Fonte: Uberti, 2012

4.1.3 DataChannels

DataChannels

Além de voz e vídeo, o projeto WebRTC também foi pensado para permitir a transmissão de dados com o intuito de permitir um variado tipo de funcionalidades como:

- Compartilhamento de tela;
- Transferência de arquivos;
- Chat/ Mensagens instantâneas;
- Jogos;
- Área de trabalho remota;
- Ensino a distancia;

A API *DataChannels* (BERGKVIST, 2011a) foi pensada para casos como esses. A expectativa é que utilizando *DataChannels* um usuário do WebRTC consiga ótimas taxas de transmissão de dados, porém ainda há alguns pontos que não estão totalmente definidos, como por exemplo o controle de banda para que a transmissão de dados não prejudique o tráfego de áudio e vídeo.

Assim como o áudio e vídeo a transmissão de dados também é feita através de *peer-to-peer*, sendo que uma característica dessa forma de transmissão de dados é que ela pode ser feita com ou sem a utilização de áudio e/ou vídeo em tempo real.

4.2 WebRTC C++ API

Esta API (BERGKVIST, 2011b) é direcionada exclusivamente para os desenvolvedores de navegadores, para que eles possam realizar a integração de seus produtos com os padrões WebRTC. Desenvolvedores de aplicações web em geral não precisam lidar com ela, todas as interações entre navegador e aplicação são feitas através das APIs previamente listadas.

4.3 Voice Engine

É o mecanismo do WebRTC, responsável pelo controle da voz, captura do áudio da placa de som e envio deste para a interface de rede. Também é responsável pelo tratamento do áudio, com os *codecs*, cancelamento de eco e redução de ruído.

4.3.1 Codecs de Audio

Segundo Bran (2013), que está tratando da utilização dos codecs de áudio no projeto WebRTC, é de grande importância que seja definido um conjunto mínimo de codecs que sejam suportados por todas as aplicações que utilizem o WebRTC para que desta forma, seja possível a interoperabilidade entre todos os clientes WebRTC. Porém nada impede que para uma aplicação mais específica, seja utilizado algum codec opcional.

No início do projeto, vários codecs foram analisados, como G.722, iSAC e iLBC porém atualmente os codecs de áudio tidos como obrigatórios segundo o draft (BRAN, 2013) que trata deste assunto são:

- Opus - com qualquer valor de tempo de pacote até 120ms;
- G.711 PCMA e PCMU com taxa de amostragem de 8KHz e 20ms de tempo de pacote;

Nos dois casos listado acima, o *draft* recomenda uma taxa de amostragem de ao menos 8KHz

Abaixo serão listadas algumas características dos codecs selecionados e em seguida será feita uma comparação entre eles.

- G.711 - Também chamado de PCM (*Pulse Code Modulation*), é um padrão ITU(*International Telecommunication Union*) lançado em 1972 que não realiza compressão da voz, apenas amostra e faz a quantização nos intervalos definidos.

Características:

- Frequência de amostragem: 8KHz;
 - Bits de quantização: 8;
 - Taxa: 64Kbps;
 - Tempo de pacote: 20ms;
- Opus - É um codec definido pela RFC 6716 e foi escolhido para o projeto WebRTC devido à sua flexibilidade e ao fato de ser totalmente livre de royalties (OPUS, 2012).

Características:

- Frequência de amostragem: 8KHz até 48KHz;
- Bits de quantização: 8;
- Taxa: 6kbps até 510kbps;
- Tempo de pacote: 2.5 até 120ms;
- Permite utilização de taxas constantes ou variáveis.

Podemos ver na Figura 4.5 um comparativo entre vários codecs disponíveis, é possível observar que o Opus tem uma excelente qualidade em praticamente todas as faixas de banda analisadas, isso faz com que ele seja em muitos casos a melhor opção quando se busca qualidade de áudio, principalmente quando estivermos tratando de um cenário entre endpoints com WebRTC.

Já o G.711, é um codec amplamente difundido e utilizado em várias tecnologias, a adoção dele no projeto WebRTC facilita em muito a interoperabilidade com outros tipos de cenários já que desta forma torna-se muitas vezes dispensável a utilização de um gateway para fazer o *transcoding* do áudio, ou seja, convertê-lo de Opus para G.711.

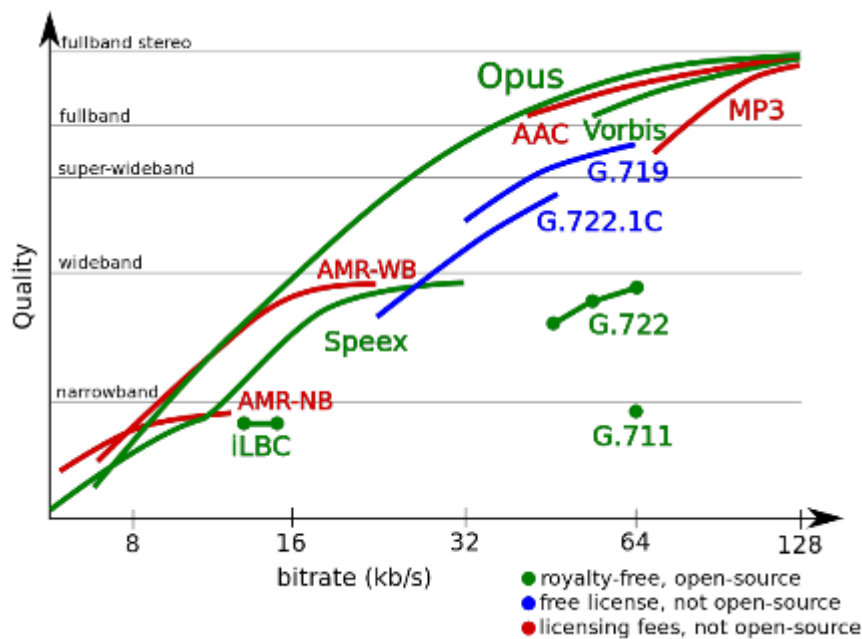


Figura 4.5: Gráfico Comparativo de Codecs

4.4 Video Engine

É a ferramenta que permite que a imagem capturada na *WebCam* seja enviada através da *web*. Utiliza o *codec VP8* e tem algumas funcionalidades como buffer de *jitter* dinâmico e uma ferramenta capaz de melhorar a imagem retirando ruídos capturados pela *WebCam*.

4.4.1 Codecs de Vídeo

No que diz respeito a escolha do codec de vídeo, conforme citado anteriormente, ainda não existe uma definição sobre qual codec será utilizado, nem ao menos foi decidido se será utilizado apenas um ou mais de um codec. O que está encaminhado é que a escolha deve ficar entre VP8 e H.264, porém uma grande disputa comercial em torno deste assunto, o que inclusive pode ser relatado como uma das causas do atraso sofrido pelo projeto até aqui.

- VP8

O *codec* de vídeo VP8 (BANKOSKI, 2011) é a evolução dos codecs VP6 (utilizado no *flash player*) e VP7 (utilizado no Skype) e foi escolhido para o projeto WebRTC por ter sido criado pela On2 Technologies, empresa que foi comprada pelo Google em 2010. Após a aquisição da On2, o código do VP8 foi aberto e liberado de *royalties* para a utilização em projetos *open source* como o WebRTC.

- H.264

O H.264(ITU-T, 2003) é definido pelo ITU-T e é um dos *codecs* mais utilizados nos dias de hoje em vários segmentos, sendo o mais utilizado na distribuição de vídeos em alta definição.

- Comparação

Em testes realizados, o H.264 geralmente se sai melhor que o VP8 quando se trata de qualidade, porém na maioria das vezes a diferença é mínima. Outro ponto a favor do H.264 é o fato de que muitos processadores de vídeo tem *hardware* dedicado para decodificá-lo, o mesmo não ocorre com VP8, onde na grande maioria dos casos, a decodificação terá de ser feita via software. Este fator é mais importante em dispositivos móveis, já que decodificação via *hardware* é mais eficiente, fazendo com que o processador principal do sistema não precise trabalhar tanto, economizando bateria. A favor do VP8 pesa principalmente o fato de ser livre de *royalties*, sendo que essa é uma premissa do WebRTC desde o seu princípio.

4.5 Sinalização

O WebRTC em si atua como um *Media Engine*, deixando a sinalização dos recursos (seja áudio, vídeo ou dados) a cargo do desenvolvedor. É ele quem escolhe qual sinalização usar, tornando desta maneira a aplicação mais flexível.

Neste capítulo serão apresentadas as sinalizações que vem sendo mais citadas como opções e que devem ser mais utilizadas nos projetos que utilizam WebRTC, são elas:

- SIP
- XMPP / Jingle
- JSEP

A seguir, estas sinalizações serão detalhadas.

4.5.1 SIP

O protocolo SIP (*Session Initiation Protocol*) (ROSENBERG, 2002), é um protocolo mantido pelo IETF que como o próprio nome diz, que tem como objetivo a inicialização de sessões.

Um sessão pode ser descrita como uma troca de informações entre uma associação de participantes. Em sua versão original é definido pela RFC 3261, sendo que há uma série de outras RFCs que a complementam e permitem diversas funcionalidades.

Para que a operação do SIP seja possível, com todos os modos e recursos previstos na sua arquitetura, são necessários alguns servidores (lógicos ou físicos) que dividirão as tarefas a serem realizadas. Apesar da divisão dos servidores, é possível que todos estejam presentes no mesmo *Hardware*.

- SIP Proxy - Um Servidor Proxy tem como função o encaminhamento das mensagens SIP entre o usuário chamador e o usuário chamado. Desta forma, ele nunca criará uma nova requisição, com exceção do método CANCEL, que permite com que uma comunicação seja cessada. Este servidor também não é responsável pelo tráfego de mídia, neste cenário este tipo de tráfego ficará entre os dois *endpoints*.
- SIP Redirect - É responsável por informar a um usuário chamador que o seu destino não pode ser acessado através do SIP *proxy* contatado.
- SIP Registrar - Responsável pela procedimento de registro de usuários através de nome de usuário e senha e a associação do mesmo a um endereço IP.
- Métodos

O protocolo SIP funciona de forma semelhante ao HTTP *Hypertext Transfer Protocol* (BERNERS-LEE, 1999), no modelo requisição/resposta. Existem diferentes métodos (requisições) para diferentes funções, seguem alguns exemplos:

- Invite - Método para início de sessão;
- Ack - Confirmação de recebimento;
- Bye - Término de sessão;
- Cancel - Cancela métodos anteriores;
- Options - Requisita quais métodos são suportados, também utilizado como *keep alive*;
- Register - Método com as informações de registro;
- Info - Sinalização durante uma chamada;
- Prack - Confirmação de resposta não definitiva;
- Outros - Há uma série de outros métodos definidos por outras RFCs;

- Respostas

Assim como o HTTP, o SIP provê diversas respostas para as requisições realizadas, a seguir a lista com as famílias de respostas possíveis:

- 1XX - Indica que a requisição foi recebida e está sendo processada;
- 2XX - Respostas que indicam sucesso;
- 3XX - É necessário algum tipo de redirecionamento para completar a requisição;
- 4XX - Erro no cliente, pode indicar requisição mal formada;
- 5XX - Falha no *server*, apesar da requisição estar no formato adequado;
- 6xx - Falhas globais;

4.5.2 XMPP/Jingle

A especificação XEP-0166(LUDWIG, 2009), define uma extensão do protocolo XMPP(SAINT-ANDRE, 2004) com a função de iniciar e gerenciar sessões de mídia entre duas entidades XMPP, com capacidade de suportar diversos tipos de mídia, como áudio, vídeo e dados.

A figura 4.6, mostra a sinalização básica trocada entre duas entidades através de Jingle, com o estabelecimento e finalização de uma sessão de mídia.

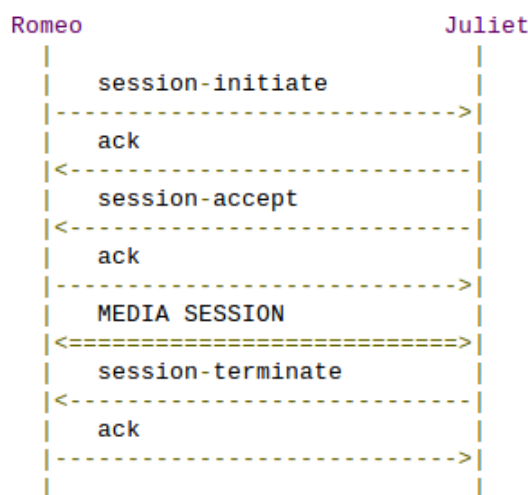


Figura 4.6: MediaStream

Fonte: XEP-0166, 2009

Além da especificação XEP-0166, existem outros documentos complementares que tratam dos métodos de transporte mídia, transporte de mídia seguro através de TLS entre outros.

4.5.3 JSEP

JSEP (*JavaScript Session Establishment Protocol*) pode ser definido como uma nova sinalização, que está sendo desenvolvida exclusivamente para o projeto WebRTC por Uberti (2012b). Trata-se de um *draft* que tem como objetivo criar um padrão que permita que uma aplicação em Javascript tenha o controle total sobre a sinalização de uma sessão multimídia sem que pra isso seja necessário implementar SIP, JINGLE ou qualquer outro método de sinalização mais complexo. Se levarmos em conta que a adoção de desse tipo de sinalização, muitas vezes pode não ser necessária, dependendo da finalidade da aplicação. Portanto a criação de uma forma mais simples de sinalização, converge com objetivo do WebRTC de permitir a criação de aplicações da maneira mais prática possível.

A descrição de mídia ainda está sendo definida, mas a princípio o JSEP deverá utilizar o protocolo SDP (*Session Description Protocol*), assim como SIP.

Outras Opções

Conforme citado anteriormente, fica à cargo do desenvolvedor decidir qual sinalização será utilizada, sendo padrões já estabelecidos como H.323, ROAP ou até mesmo protocolos de sinalização proprietários.

4.6 Linguagens necessárias ao projeto.

Para que seja possível analisar corretamente a arquitetura do projeto WebRTC, é preciso se ater à algumas tecnologias por trás deste projeto, as quais sem elas, o mesmo não seria possível. Uma das novidades que tornou a criação do WebRTC viável foi a utilização da linguagem HTML5, que trouxe uma série de novidades em relação as suas versões anteriores, em conjunto com a linguagem Javascript, essas evoluções foram determinantes para que o desenvolvimento do WebRTC fosse possível.

4.6.1 HTML5

O HTML *HyperText Markup Language* (HICKSON, 2013) é uma linguagem de marcação de hipertextos utilizada para entrega de conteúdos na web. Ela foi criada a partir de 1990, sendo que em seus primeiros cinco anos de desenvolvimento passou por inúmeras revisões e teve muitas extensões.

Este tipo de documento pode ser criado utilizando qualquer editor de texto e contém elementos rodeados de tags. As tags são as marcações utilizadas para inclusão de hyperlinks, imagens e outras mídias na web.

Dentro de um site o HTML é o responsável por abranger todo o conteúdo com tags que fornecem informações sobre o conteúdo e a natureza além de referências a imagens e outras mídias.

A versão mais recente do HTML é a versão 5 (BERJON, 2013), que trouxe um conjunto de inovações que são indispensáveis para o WebRTC como por exemplo as tags de Vídeo e Áudio.

4.6.2 Javascript

O JavaScript (ECMA, 2011) é uma das linguagens utilizadas para acessar partes de um documento HTML. Ela é chamada de linguagem de Script, pois se diferencia das demais que são compiladas. O que a muda é a forma como ela traduz as informações de Script, onde as compiladas traduzem de imediato antes de utilização e a de script é interpretada linha a linha pelos navegadores. Ele foi desenvolvido por Brendan Eich e lançado pela primeira vez na versão beta no navegador Netscape 2.0 em 1995. Ela tem se tornado a linguagem de programação mais popular da web se tornando hoje a principal para programação client-side em navegadores web.

Pode-se compreender o Java da seguinte forma: em uma página da web, o HTML é utilizado para armazenar o conteúdo e a formatação da página, o CSS (folhas em estilo cascada) codifica o conteúdo de forma a definir como ele se apresentará graficamente e o JavaScript é o responsável por criar efeitos mais ricos ou aplicações web ricas. Ele é utilizado para tornar os sites mais dinâmicos e interativos.

O fato do Javascript ser processado localmente no navegador, ajuda a permitir as funções *peer-to-peer* que o WebRTC implementa.

4.6.3 Segurança

Segundo Dutton (2013), existem várias maneiras de uma aplicação de comunicação em tempo real nos formatos atuais, comprometerem a segurança:

- Interceptação de tráfego não criptografado;
- Uma aplicação pode realizar gravações do áudio e vídeo sem o conhecimento do usuário;
- Instalação de softwares mal intencionados junto com os plugins necessários para a comunicação;

- Falhas de segurança nos plugins utilizados;

Tendo isso em mente, a questão segurança está presente desde o começo da definição dos padrões do WebRTC, e há um draft(RESORLA, 2013) no IETF apenas discutindo estas questões. Para evitar problemas como os listados, o WebRTC faz uso de SRTP (*Secure Real-time Transport Protocol*, protocolo definido na RFC 3711(BAUGHER, 2004) que tem como principal função criptografar a mídia, para que esta não possa ser decodificada por terceiros.

Outra questão que o WebRTC se preocupa, é com o acesso as fontes de mídia do usuário, como câmera e microfone, também pra evitar que alguém mal intencionado ative estas ferramentas sem a permissão do usuário. Para combater este problema, sempre que alguma aplicação requisita a utilização da câmera/microfone de algum usuário, este deve aceitar através de um prompt que é exibido, como mostra a Figura 4.7.

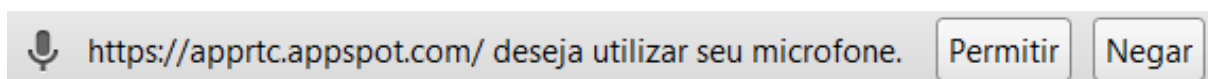


Figura 4.7: Autorização para utilização de recursos

Enquanto o usuário não permitir, nenhuma informação é capturada das entradas de mídia. Além disso enquanto microfone/câmera estiverem em uso, é mostrada uma notificação na barra de notificações do sistema operacional bem como na aba do navegador que está usando esta função.

4.6.4 Navegadores

Os principais desenvolvedores já estão adotando o WebRTC como solução nativa de seus navegadores. Browsers como Google Chrome, Mozilla Firefox e Opera, já suportam parcialmente as funcionalidade do WebRTC, e em breve devem suporta-lo por completo.

A seguir, está a lista dos estágios de desenvolvimento de alguns navegadores em relação ao WebRTC:

Chrome

- *MediaStreams*: A partir da versão 21;
- *PeerConnection*: A partir da versão 22;
- *DataChannels*: Suporte básico na versão 25, devendo ser ampliado na versão 26;

Firefox

- *MediaStreams*: A partir da versão 17;
- *PeerConnection*: A partir da versão 18;
- *DataChannels*: A partir da versão 18 com restrições;

Opera

A versão 12 do Opera já disponível, suporta WebRTC porém apenas *MediaStreams*.

Comparação entre os navegadores de desktops

A Figura 4.8 faz uma comparação do suporte dado ao WebRTC pelos navegadores *Web* mais utilizados no mundo. As versões marcadas em verde indicam quando os navegadores passaram a suportar o projeto, mesmo que parcialmente.

| Show all versions | IE | Firefox | Chrome | Safari | Opera |
|-------------------|------|---------|--------|--------|-------|
| | 8.0 | | | | |
| | 9.0 | 18.0 | 24.0 | 5.1 | |
| Current | 10.0 | 19.0 | 25.0 | 6.0 | 12.1 |
| Near future | | 20.0 | 26.0 | | 12.5 |
| Farther future | | 21.0 | 27.0 | | |

Figura 4.8: Suporte nos navegadores de Desktop

Fonte: *caniuse.com*, 2013

Pode-se ver que tanto o *Internet explorer* quanto o *Safari* ainda não tem suporte ao WebRTC, e não há indícios de que isso seja alterado no curto prazo.

Navegadores de Dispositivos Móveis

O primeiro navegador a suportar WebRTC nas plataformas móveis foi o *Bowser*, desenvolvido pela Ericsson justamente com o objetivo possibilitar a realização de testes de aplicações WebRTC utilizando Smartphones

Até o momento o único navegador a suportar WebRTC em dispositivos móveis é o *Bowser*, desenvolvido pela Ericsson que está disponível para Android e iOS, porém ele não ainda suporta todas as funcionalidades, limitando desta forma o uso das aplicações mais comuns.

A expectativa é de que em breve teremos WebRTC também nos principais navegadores para smartphones e tablets. Assim que o sistema estiver estável nas versões para desktop, deve ser portado aos navegadores de plataformas móveis.

Interoperabilidade entre Navegadores

Uma das questões cruciais para que o WebRTC se torne um padrão amplamente adotado é a interoperabilidade entre os diferentes navegadores. O fato de dois navegadores suportarem o protocolo não garante que eles consigam falar entre si. O Primeiro movimento nesta direção foi apresentado no dia 30 de janeiro de 2013 onde engenheiros da fundação Mozilla e do Google anunciaram a interoperabilidade entre os navegadores Firefox e Chrome através de um video lançado na página oficial do projeto¹, porém até o momento, esta interação é possível apenas nas versões de teste destes dois navegadores.

¹<http://www.webrtc.org/demo/>

5 *Cenários e Testes*

5.1 Cenários

Devido a forma como foi concebido, o WebRTC permite uma vasta quantidade de cenários distintos já que possibilita uma grande variedade de elementos se comunicando através de Servidores Web (JOHNSTON, 2012). A Figura 5.1 mostra essa diversidade.

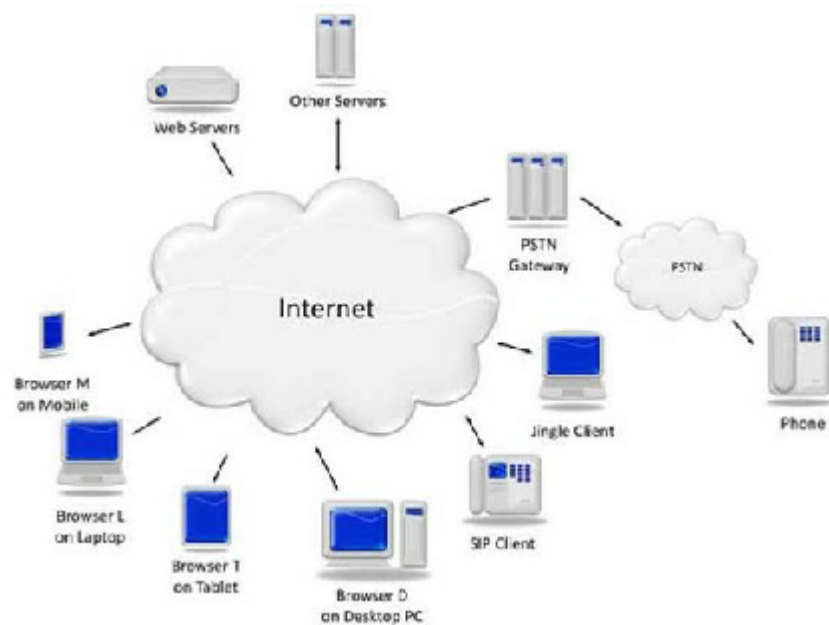


Figura 5.1: Relação entre Browser e WebServer

Fonte: Jhonston, 2012

Podem ser elementos de uma rede WebRTC:

- Navegadores para dispositivos móveis (*Smartphones e Tablets*);
- Navegadores de PCs;
- Servidores Web;
- Gateways para outras sinalizações;

Serão mostrados alguns exemplos de cenários possíveis com estes elementos na seções a seguir.

5.1.1 Comunicação entre navegadores num mesmo WebServer

Esta é a forma mais básica de utilização do WebRTC, onde dois navegadores com suporte a essa tecnologia trocam mensagens através de um WebServer e fazem uma conexão *peer-to-peer* para transmissão da mídia (áudio/vídeo). como mostra a Figura 5.2.

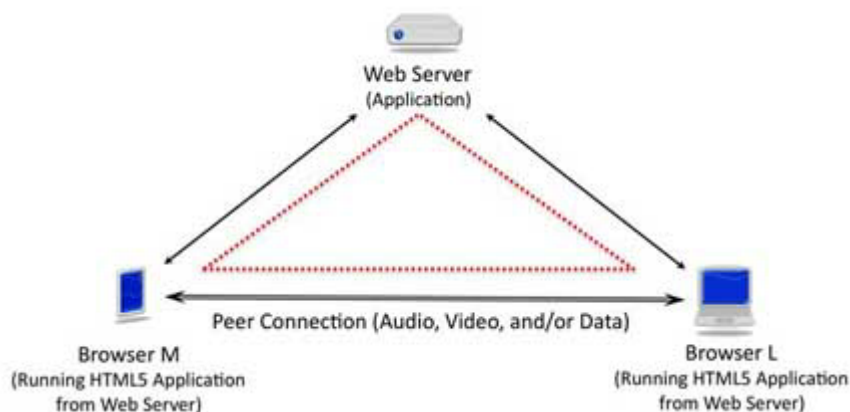


Figura 5.2: Comunicação entre dois Navegadores usando WebRTC através de um *WebServer*

Fonte: Jhonston, 2012

Uma aplicação que implementa este tipo de cenário é a Apprtc¹, que é tida como a aplicação referência do WebRTC pois possui o cenário mais básico possível. Ela permite que dois usuários realizem uma videochamada através de um portal *Web*.

Quando um usuário acessa a página, ele recebe um endereço com um código de 8 dígitos, basta que outra pessoa entre no endereço utilizando o mesmo código para que a chamada seja estabelecida. A figura 5.3 mostra esta solução em funcionamento.

Uma das curiosidades desta aplicação é que até o momento, ela é a única capaz de prover a comunicação entre navegadores de plataforma móveis como por exemplo o Chrome Beta, e navegadores instalados em *Desktops*.

5.1.2 Comunicação entre navegadores com WebServers Distintos

Este é um cenário semelhante ao apresentado na seção anterior, porém a com diferença da presença de dois *WebServers* diferentes. Neste caso os Servidores estariam se comunicando

¹<https://apprtc.appspot.com/>

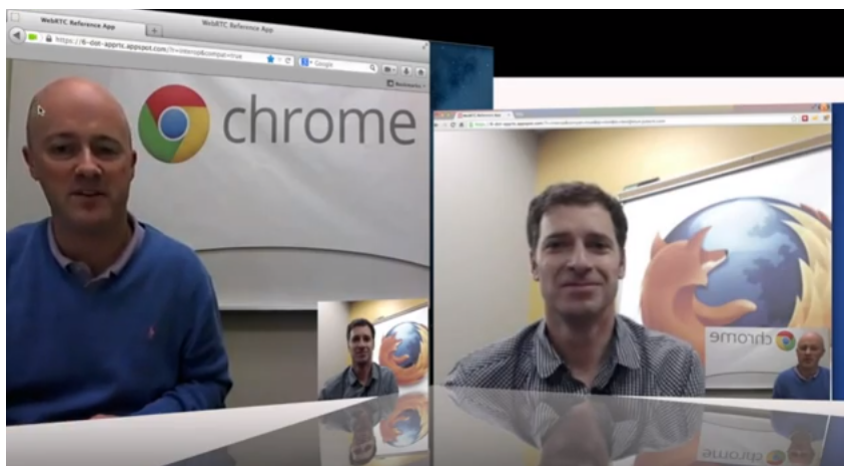


Figura 5.3: Apprtc em funcionamento

através de alguma sinalização específica com a finalidade de prover interoperabilidade entre por exemplo, aplicações distintas, como mostra a figura 5.5.

Foram feitas várias tentativas de implementar um cenário como esse, utilizando dois *web-servers*, mas infelizmente não foi obtido sucesso em nenhuma delas. Nos testes foram utilizadas máquinas virtuais rodando o código fonte da aplicação (AppRTC), onde detectamos que, as aplicações atuais para esta topologia, são limitadas a cenários muito específicos, portanto no teste que fizemos em uma rede local elas não se comportaram da forma esperada. Porém o resultado teórico seria semelhante ao apresentado na Figura 5.4, com as trocas de mensagens HTTP entre o servidor e os navegadores

5.1.3 Integração com outras tecnologias

Uma das circunstâncias mais comuns que devem aparecer quando o WebRTC começar a ser mais amplamente utilizado, será a integração com os cenários já existentes que não podem ser simplesmente ignoradas ou trocadas para a adoção da nova tecnologia. Deste modo é fundamental a utilização de *gateways* com o intuito de converter o WebRTC em um formato que seja compreendido por estas outras tecnologias.

5.1.4 Integração com SIP

A figura 5.7 mostra de forma simples como seria a comunicação entre um Navegador utilizando WebRTC e um cliente SIP ligado a um *SIP Server*, onde o *WebServer* teria a função de ser um *gateway* entre WebRTC e SIP para desta forma se comunicar com o *SIP Server*.

Este layout pode ser utilizado com SIP ou com qualquer outra sinalização que o desenvol-

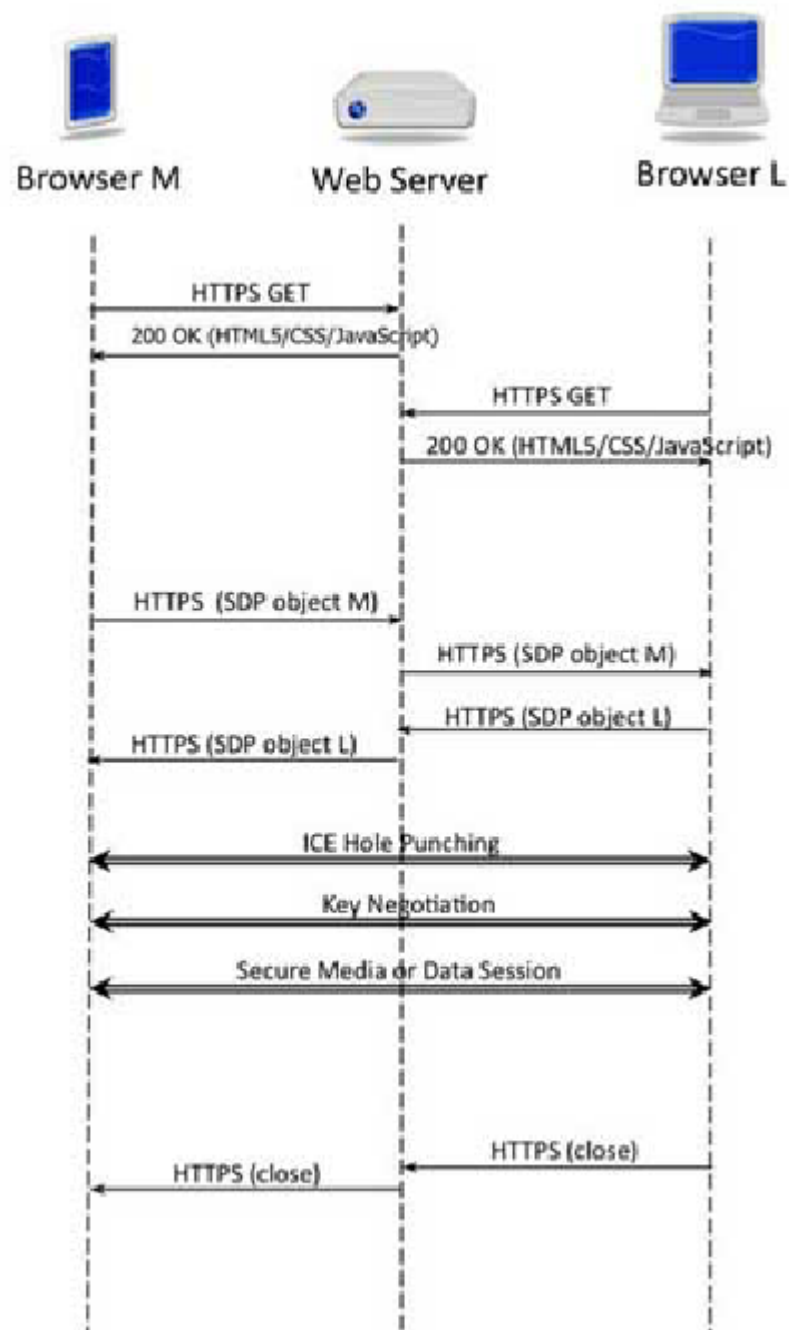


Figura 5.4: Troca de mensagens entre Navegadores e WebServer

Fonte: Jhonston, 2012

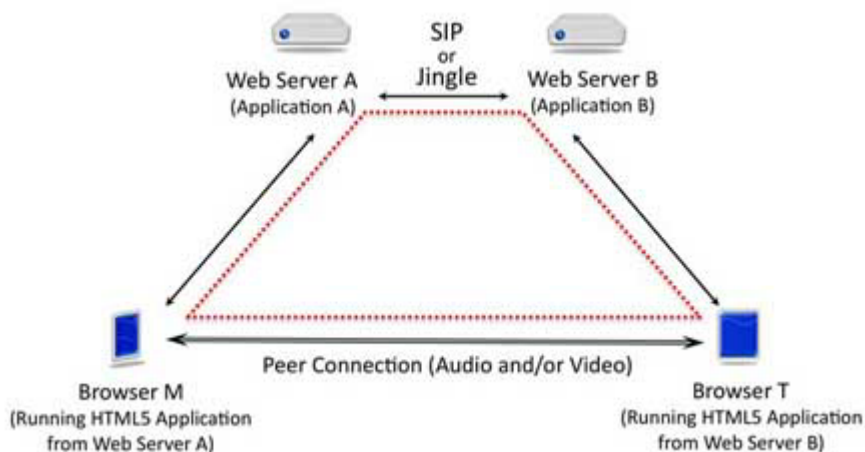


Figura 5.5: Dois navegadores usando servidores distintos

Fonte: Jhonston, 2012

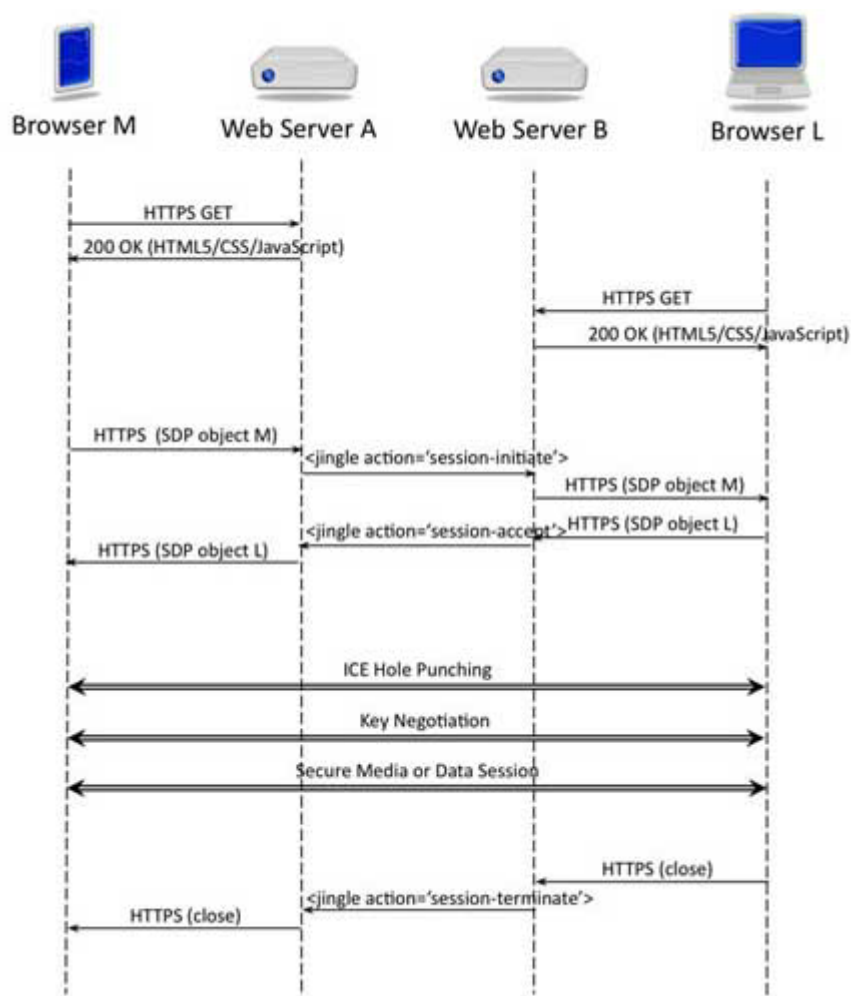


Figura 5.6: Troca de mensagens entre Navegadores e WebServer

Fonte: Jhonston, 2012

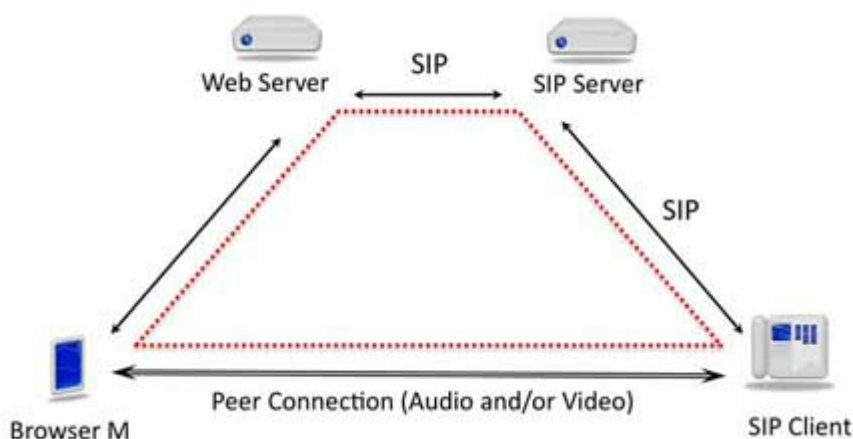


Figura 5.7: WebRTC com SIP

vedor desejar, seja ela Jingle, H.323 ou até mesmo uma que seja proprietária.

Este é um cenário semelhante a muitos casos práticos, incluindo por exemplo a rede de telefonia do IFSC campus São José. Por esse motivo, foi elaborado um cenário de testes com o objetivo de demonstrar como seria uma possível implementação do WebRTC se comunicando com a rede VoIP já existente, sem que para isso fosse preciso alterar os elementos já presentes na rede..

O cenário consiste em :

- Um computador com um navegador que tenha suporte ao WebRTC;
- Um servidor *Web* rodando a aplicação SIPML5;
- Um servidor *Web* rodando a aplicação WebRTC2SIP;
- Um PABX com suporte ao protocolo SIP;
- Uma terminação registrada no PABX IP, como por exemplo um telefone IP ou *Softphone*;

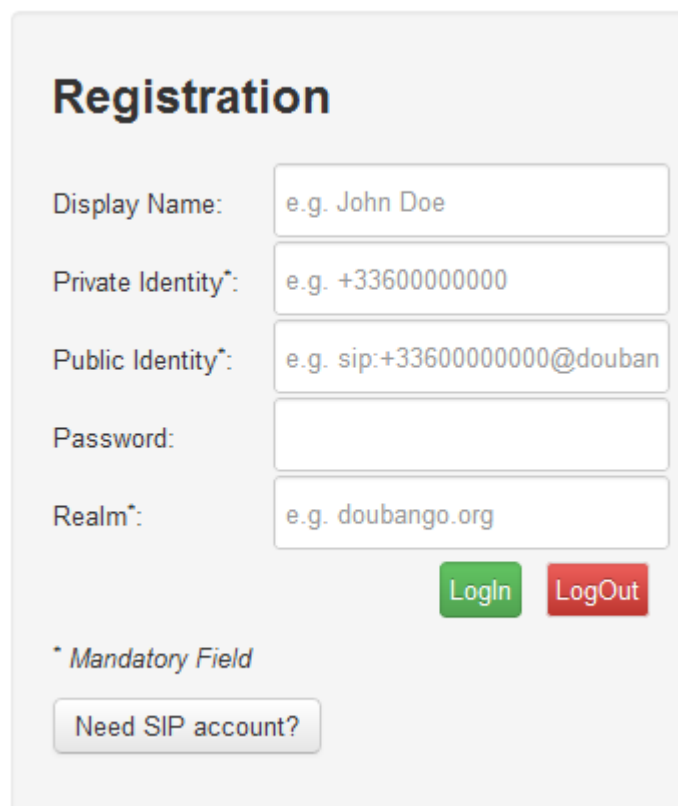
A seguir detalhamos algumas das tecnologias citadas acima:

SIPML5

O SIPML5 é uma aplicação desenvolvida pela Doubango Telecom¹, que implementa a pilha do protocolo SIP em Javascript. Conforme citado anteriormente, a sinalização é uma escolha do desenvolvedor da aplicação. A escolha do SIPML5 para este teste, deve-se ao fato da sua semelhança com um *softphone* comum, como mostra a Figura 5.8, ele é bem simples de ser

¹<http://www.doubango.org/>

configurado, desta forma a migração de usuários já familiarizados com este tipo de aplicação é facilitada.



The image shows a web form titled "Registration" for creating a SIPML5 account. It contains several input fields with placeholder text: "Display Name" (e.g. John Doe), "Private Identity*" (e.g. +33600000000), "Public Identity*" (e.g. sip:+33600000000@douban), "Password", and "Realm*" (e.g. doubango.org). There are "Login" and "Logout" buttons. A note indicates that fields with an asterisk are mandatory. At the bottom, there is a button labeled "Need SIP account?".

Figura 5.8: Configuração de uma conta com o SIPML5

Gateway WebRTC2SIP

O WebRTC2SIP (TELECOM, 2012) é um gateway que foi criado com o objetivo de prover a interoperabilidade entre soluções que utilizam o WebRTC com sistemas que tem como base o SIP. Ele é responsável por realizar a conversão dos pacotes enviados pelo endpoint que utiliza webRTC, no caso o SIPML5, para o PABX IP.

O WebRTC2SIP também tem a capacidade de converter o protocolo de transporte de mídia, através da função RTCWEB Breaker. Ele converte os pacotes SRTP, obrigatórios no WebRTC, para pacotes RTP em situações onde o PABX IP não está configurado para receber os pacotes RTP seguros. Um diagrama completo do WebRTC2SIP é exibido na Figura 5.9.

PABX IP

Para este teste, utilizamos como PABX o Asterisk (DIGIUM, 2013), que é um software de código aberto para a criação de aplicações de comunicação, como PABX IP, URAs, discadores,

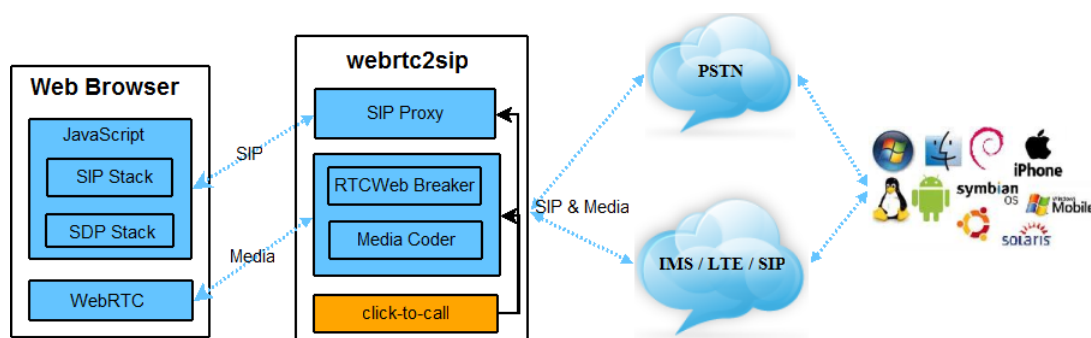


Figura 5.9: Diagrama WebRTC2SIP

etc. É válido ressaltar, que a partir da versão 11, o Asterisk tem suporte nativo ao WebRTC, dispensando o uso de um gateway como é o caso do WebRTC2SIP, porém como o objetivo deste teste é mostrar a interoperabilidade entre WebRTC e um PABX IP genérico, estes recursos de suporte ao WebRTC não foram utilizados.

A topologia foi montada com a utilização de máquinas virtuais, com objetivo de simular uma rede de verdade, o cenário ficou como mostra a imagem 5.10.

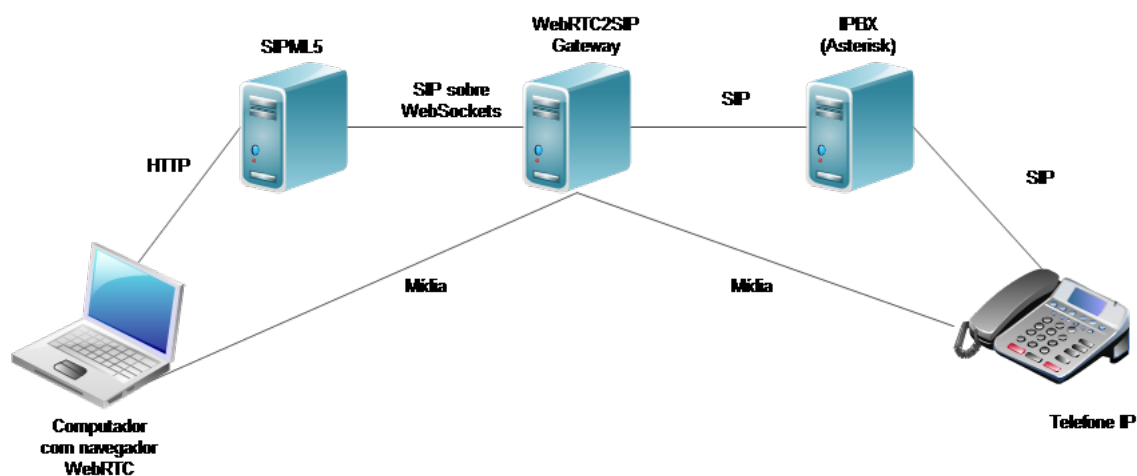


Figura 5.10: Cenário de testes

Apesar de alguns problemas na configuração inicial do WebRTC2SIP, os testes seguiram sem maiores problemas, com chamadas sendo feitas nos dois sentidos e obtendo boa qualidade de áudio. Lembrando que não foi preciso realizar nenhuma alteração no asterisk para que as chamadas fossem completadas.

A troca de mensagens entre WebRTC2SIP e Asterisk é mostrada na figura 5.11. A primeira mensagem enviada pelo cliente WebRTC é equivalente a um registro, ou seja, o usuário está registrado no Asterisk como outro *endpoint* qualquer. A segunda mensagem entre o cliente WebRTC e o WebRTC2SIP (WebServer) é o início da chamada, após isso temos o atendimento, a troca de mídia e desconexão.

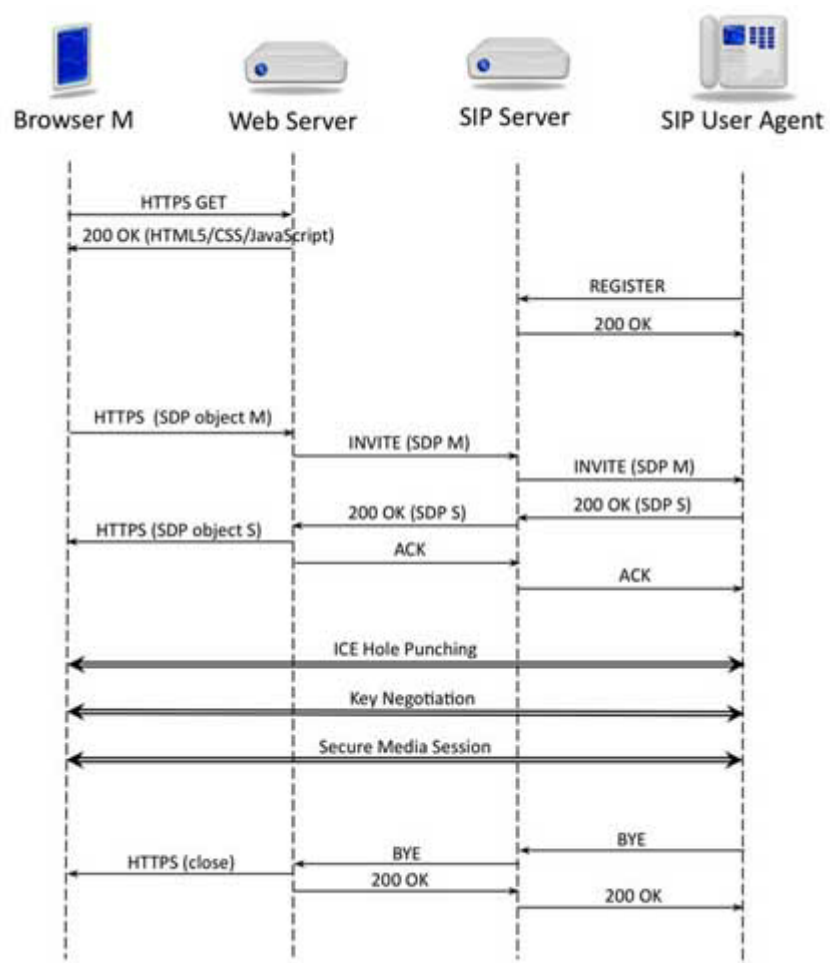


Figura 5.11: Troca de mensagens WebRTC operando em conjunto com SIP

Do lado do Asterisk temos uma troca de sinalização normal como qualquer chamada SIP. A Figura 5.12 mostra a sinalização exata entre WebRTC2SIP e Asterisk, gerada a partir de uma captura realizada neste cenário.

| Time | 192.168.1.111 | 192.168.1.112 | Comment |
|--------|--------------------------------|-------------------|--|
| 7,728 | INVITE SDP (g711U g711A teleph | (10060) → (5060) | SIP From: <sip:webrtc@192.168.1.101 To:<sip:200@asterisk |
| 7,729 | 401 Unauthorized | (10060) → (5060) | SIP Status |
| 7,754 | INVITE SDP (g711U g711A teleph | (10060) → (5060) | SIP From: <sip:webrtc@192.168.1.101 To:<sip:200@asterisk |
| 7,755 | 100 Trying | (10060) → (5060) | SIP Status |
| 8,281 | 180 Ringing | (10060) → (5060) | SIP Status |
| 10,753 | 200 OK SDP (g711U g711A teleph | (10060) → (5060) | SIP Status |
| 10,763 | ACK | (10060) → (5060) | SIP Request |
| 10,841 | RTP (g711U) | (44790) → (13672) | RTP Num packets:381 Duration:7.589s SSRC:0x4E2A5CF5 |
| 10,953 | RTP (g711U) | (44790) → (13672) | RTP Num packets:375 Duration:7.475s SSRC:0x631CF422 |
| 18,441 | BYE | (10060) → (5060) | SIP Request |
| 18,441 | 200 OK | (10060) → (5060) | SIP Status |

Figura 5.12: Troca de mensagens WebRTC2SIP e Asterisk

5.1.5 Integração com Telefonia Fixa

Um dos cenários que pode ser mais utilizado no WebRTC é a sua integração com a rede de telefonia fixa comutada, assim como é feito hoje com a telefonia VoIP “convencional” onde em muitos casos a chamada iniciada num ambiente VoIP é terminada em um ambiente STFC (Serviço de Telefonia Fixa Comutada). A Figura 5.13 ilustra bem este cenário.

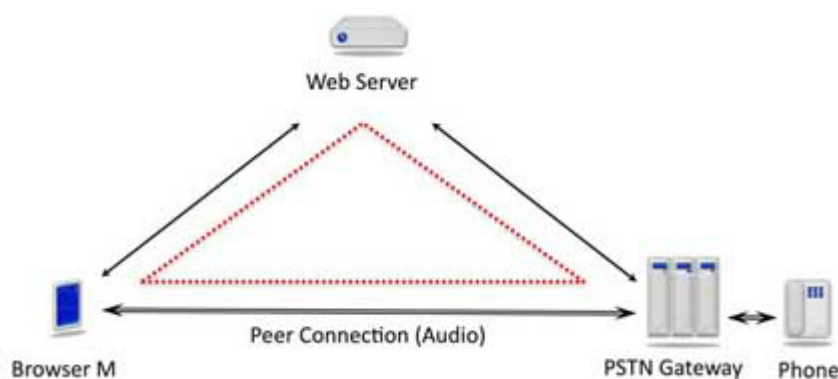


Figura 5.13: WebRTC com telefonia fixa

Este é um cenário que ainda inviável até o momento, já que não existe nenhuma aplicação que faça uma conversão direta entre WebRTC e telefonia comutada. Entretanto, poderia ser utilizado um cenário semelhante ao que foi testado na seção 5.1.4, com a diferença de que

poderia ser utilizado um Asterisk com placas para telefonia ou qualquer outro *gateway* SIP e o PSTN

6 *Conclusões*

Este trabalho teve como objetivo analisar e testar o novo padrão de comunicação em tempo real WebRTC, de modo a verificar que inovações este projeto traz e quais as funcionalidades possíveis de serem implementadas pelos desenvolvedores em aplicações baseadas nele.

Apesar do foco do trabalho não ser o desenvolvimento de nenhuma aplicação baseada em WebRTC, e sim o estudo da tecnologia, durante o estudo teórico, foi possível constatar a facilidade no uso das APIs do projeto, o que torna o desenvolvimento de aplicações simples se comparado a todo o trabalho que seria necessário se o desenvolvedor tivesse que lidar com tratamento de mídia, interação com o sistema operacional para a captura das mídias, questões de segurança, entre outros. Essa facilidade vem fazendo com que um grande número de *startups*² esteja adotando o WebRTC como base de suas soluções.

O fato da sinalização ser uma escolha do desenvolvedor, confere ao WebRTC um grande flexibilidade, fazendo com que possa ser utilizado em uma grande diversidade de cenários. Um dos principais objetivos do trabalho era implementar alguns destes cenários para a realização de testes e ver o comportamento das ferramentas, além da qualidade entregue ao usuário.

Infelizmente, o padrão WebRTC ainda não está em uma fase madura, o que dificultou e até inviabilizou muitos dos testes que este trabalho tinha a intenção de realizar. O teste de maior importância era verificar como seria possível integrar uma solução utilizando o WebRTC a um cenário de telefonia semelhante ao IFSC câmpus São José, sem que pra isso nada precisasse ser alterado na atual estrutura de telefonia. Esse teste, apesar de ser um dos cenários mais complexos que poderia ser analisado, foi onde obtivemos o melhor resultado, mostrando que seria totalmente possível a integração da tecnologias WebRTC com um cenário similar ao do IFSC apenas adicionando alguns serviços como o SIPML5 e o WebRTC2SIP, sem que para isso fosse preciso realizar qualquer alteração no cenário existente. Isso mostra mais uma vez a flexibilidade trazida pelo WebRTC e sua grande capacidade de ser integrado com outras tecnologias já existentes.

²Empresas novas com o intuito de implantar novas ideias.

Outras dificuldades encontradas no decorrer do trabalho foram a falta de conteúdo sobre o assunto, onde na maioria das vezes a única informação disponível eram os drafts ainda em desenvolvimento e a constante mudança de conceitos, obviamente este comportamento era esperado por se tratar de um projeto muito recente.

Durante todos os testes realizados, seja os que foram implementados como estudo de caso ou com serviços já existentes, foi constatada uma grande qualidade de áudio e vídeo, o maior problema detectado foi o *delay* em alguns testes, que não chegou a prejudicar a comunicação, e é até compreensível se levarmos em consideração que os pacotes estavam trafegando pela internet, sem nenhuma política de priorização de tráfego específica.

Pode-se dizer que o WebRTC tem grandes chances de se tornar um padrão difundido e amplamente utilizado, devido a sua proposta e ao fato de possuir grandes empresas empenhadas em seu desenvolvimento, porém para que essa meta seja alcançada, ainda é preciso evoluir muito.

A expectativa era de que quando este trabalho estivesse para ser concluído, o projeto WebRTC já se encontraria em um estágio mais avançado, por exemplo, a implementação dos *DataChannels* estava prevista para o último trimestre de 2012. Porém apenas recentemente, foram lançados as primeiras versões de navegadores com suporte a esta função, e mesmo assim ainda são versões que a implementam parcialmente. O mesmo vale para os navegadores com suporte ao WebRTC nos sistemas operacionais para *mobile*, era esperado que tivessemos dispositivos móveis rodando WebRTC no final de 2012, porém somente no começo de março de 2013 foi disponibilizada uma versão beta do Chrome que suporta o projeto.

Boa parte dos atrasos no andamento do projeto deve-se a problemas comerciais, como a disputa entre VP8 e H.264. Para que o projeto evolua da maneira esperada, é necessário que esses problemas sejam resolvidos o mais brevemente possível, e que sejam tomadas as decisões baseadas principalmente no aspecto técnico.

Apesar destes problemas, o WebRTC traz aos navegadores funcionalidades até então deficientes, que só eram possíveis com a utilização de *plugins*, acarretando todo o tipo de problemas que eles podem trazer. Num mundo onde temos cada vez mais serviços rodando na nuvem, é interessante pensar nos usos que esta tecnologia pode trazer.

Em breve poderemos ter por exemplo, uma solução completa de ensino a distância baseada em WebRTC, permitindo aos usuários videoaulas, troca de arquivos entre os usuários, trabalhos colaborativos e compartilhamento de tela. Tudo isso disponível em um sistema multiplataforma, acessível de *smartphones* e *tablets*, enfim as possibilidades são inúmeras e ainda há muito que

ser explorado.

Como sugestão para trabalhos futuros, podemos citar a criação de um *gateway* para conversão direta entre WebRTC e PSTN, além de diversos outros testes que serão possíveis com a evolução do projeto, como testes de qualidade de serviço, segurança entre outros.

Referências Bibliográficas

- ALVESTRAND, H. *RTCWEB Overview*. 2013. Disponível em: <<http://tools.ietf.org/html/draft-ietf-rtcweb-overview-06>>.
- BANKOSKI, J. *RFC 6386*. 2011. Disponível em: <<http://datatracker.ietf.org/doc/rfc6386/>>.
- BAUGHER, M. *The Secure Real-time Transport Protocol (SRTP)*. 2004. Disponível em: <<http://tools.ietf.org/html/rfc3711>>.
- BERGKVIST, A. *WebRTC Datachannel*. 2011. Disponível em: <<http://dev.w3.org/2011/webrtc/editor/webrtc.html#rtcdatachannel>>.
- BERGKVIST, A. *WebRTC Native API*. 2011. Disponível em: <<http://dev.w3.org/2011/webrtc/editor/webrtc.html>>.
- BERJON, R. *HTML 5.1 Nightly*. 2013. Disponível em: <<http://www.w3.org/html/wg/drafts/html/master/>>.
- BERNERS-LEE, T. *Hypertext Transfer Protocol – HTTP/1.1*. 1999. Disponível em: <<http://www.ietf.org/rfc/rfc2616.txt>>.
- BRAN, J. V. C. *WebRTC Audio Codec and Processing Requirements draft-ietf-rtcweb-audio-01*. 2013. Disponível em: <http://datatracker.ietf.org/doc/draft-ietf-rtcweb-audio/?include_text=1>.
- CISCO. *Cisco Visual Networking Index: Forecast and Methodology, 2011-2016*. 2012. Disponível em: <http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html>.
- DIGIUM. *WebRTC2SIP*. 2013. Disponível em: <<http://www.asterisk.org/>>.
- DUTTOM, S. *Real-time communication without plugins*. 2013. Disponível em: <<http://www.html5rocks.com/en/tutorials/webrtc/basics/>>.
- ECMA. *Standard ECMA-262*. 2011. Disponível em: <<http://www.ecma-international.org/publications/standards/Ecma-262.htm>>.
- GOOGLE. *WebRTC Architecture*. 2012. Disponível em: <<http://www.webrtc.org/reference/architecture>>.
- GOOGLE. *WebRTC Web API*. 2012. Disponível em: <<http://www.webrtc.org/reference/architecture#TOC-Web-API>>.
- HICKSON, I. *HTML - Living Standard*. 2013. Disponível em: <<http://www.whatwg.org/specs/web-apps/current-work/multipage/>>.

ITU-T. *H.264 Specification*. 2003. Disponível em: <<http://www.itu.int/rec/T-REC-H.264-200305-S/en>>.

ITU-T. *H.323*. 2009. Disponível em: <<http://www.itu.int/rec/T-REC-H.323/>>.

JENNINGS, C. *RTCWeb Offer/Answer Protocol (ROAP)*. 2012. Disponível em: <<http://tools.ietf.org/html/draft-jennings-rtcweb-signaling-01>>.

JOHNSTON, A. B. *WebRTC - APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. [S.l.]: Addison-Wesley, Reading, Massachusetts,, 2012.

LUDWIG, S. *XEP-0166: Jingle*. 2009. Disponível em: <<http://xmpp.org/extensions/xep-0166.html>>.

O'CALLAHAN, R. *WebRTC Web API*. 2012. Disponível em: <<https://dvcs.w3.org/hg/audio/raw-file/tip/streams/StreamProcessing.html>>.

OPUS. *Opus Interactive Audio Codec*. 2012. Disponível em: <<http://www.opus-codec.org/>>.

RESCORLA, E. *RTCWEB Security Architecture*. 2013. Disponível em: <<http://datatracker.ietf.org/doc/draft-ietf-rtcweb-security-arch/>>.

ROETTIGERS, J. *Microsoft commits to WebRTC, just not Googles version*. 2012. Disponível em: <<http://gigaom.com/2012/08/06/microsoft-webrtc-w3c/>>.

ROSENBERG, J. *SIP: Session Initiation Protocol*. 2002. Disponível em: <<http://www.ietf.org/rfc/rfc3261.txt>>.

SAINT-ANDRE, P. *Extensible Messaging and Presence Protocol (XMPP)*. 2004. Disponível em: <<http://xmpp.org/rfcs/rfc3920.html>>.

TELECOM, D. *WebRTC2SIP*. 2012. Disponível em: <<https://code.google.com/p/webrtc2sip/>>.

TELEGEOGRAPHY. *TeleGeography Report and Database*. 2013. Disponível em: <<http://www.telegeography.com/research-services/telegeography-report-database/index.html>>.

UBERTI, J. *Google I/O 2012 - WebRTC: Real-time Audio/Video and P2P in HTML5*. 2012. Disponível em: <http://www.youtube.com/watch?feature=player_embedded&v=E8C8ouiXHHk>.

UBERTI, J. *Javascript Session Establishment Protocol*. 2012. Disponível em: <<https://tools.ietf.org/html/draft-ietf-rtcweb-jsep-02>>.

UBERTI, J. *WebRTC - real-time audio/video and p2p in html5*. In: . [S.l.]: Apresentado no evento Google IO 2012, 2012.

YORK, D. *Google buys GIPS for \$68 million*. 2010. Disponível em: <<http://www.disruptivetelephony.com/2010/05/google-buys-gips-for-68-million—to-take-on-skype-apple-microsoft.html>>.