**Tecnologia**

**Já conhece o projeto WebRTC?**

[**http://cio.com.br/tecnologia/2013/06/07/ja-conhece-o-projeto-webrtc/**](http://cio.com.br/tecnologia/2013/06/07/ja-conhece-o-projeto-webrtc/)

**O objetivo é tornar as comunicações peer-to-peer na web tão fáceis como usar um telefone. Saiba como configurar a API e as limitações que o protocolo ainda enfrenta**

**Chris Minnick e Ed Tittel, CIO/EUA**

**Publicada em 07 de junho de 2013 às 13h28**

Todo mundo já teve a experiência de tentar participar de uma Web conference, apenas para perceber que deveria primeiro ter feito o download de algum plugin, atualizar o Java ou o Flash, ou instalar outro aplicativo. Se você já teve que explicar a um cliente como ou por que ele precisa baixar e instalar um plugin ou aplicativo para encontrá-lo on-line,provavelmente também já teve a experiência desistir e dizer: "Você sabe o quê?  Vou usar o telefone e ligar para você!"

Transmissões ao vivo de áudio e vídeo e Web conferences estão mudando as comunicações empresariais e as pessoais, mas na maioria das vezes ainda são consideradas muito complicadas ou não confiáveis por muitos usuários finais. Da mesma forma, a criação de uma aplicação para comunicação em tempo real (RTC) é muito complicada para o desenvolvedor web mediano.

Tudo isso está mudando agora, graças ao projeto Comunicação Web em Tempo Real ( WebRTC ), um padrão da indústria para permitir aos navegadores executar aplicações de chamada telefônica, vídeo chat e compartilhamento P2P sem a necessidade de plugins.

**Hoje, o Real-Time Communications significa "Call Me**"

Até agora, as comunicações em tempo real necessitavam a instalação de um plugin ou um aplicativo nativo. Os usuários precisam baixar, instalar, atualizar, configurar ou solucionar vários problemas para conseguir usá-los.

Hoje, você pode usar várias aplicações de comunicação em tempo real, incluindo Skype, FaceTime, Google Talk, Yahoo Messenger, iChat, GotoMeeting e join.me. E ainda os telefones VOIP ou todos os aplicativos RTC em seus smartphones e tablets, também. Sem dúvida, existem vários outros aplicativos RTC em seus computadores que você tenha baixado e usado apenas uma vez, ou que foram baixados na forma de aplicativos em Flash ou Java quando visitou vários sites de negócios e usou seus recursos de bate-papo ao vivo.

Você também pode usar alguns clientes RTC para se comunicar com outras pessoas usando diferentes programas RTC. Do iChat para o Yahoo Messenger, por exemplo. Na maioria das vezes, no entanto, cada um destes programas é projetado para funcionar melhor, ou apenas com, outros computadores que executam o mesmo software ou plugin.

**Comunicação direto no browser**

Em contraste com este cenário, o World Wide Web Consortium (W3C) decidiu criar o  WebRTC. Qualquer pessoa com um navegador Web e um microfone pode fazer chamadas para qualquer pessoa com um navegador Web e um microfone. Se uma ou ambas as partes tiverem algum tipo de câmera de vídeo, a chamada também pode envolver vídeo.

Além disso, as APIs JavaScript envolvidas para permitir esta comunicação peer-to-peer são simples o suficiente para que você possa criar um cliente WebRTC com apenas cinco ou seis linhas de JavaScript e HTML. Os navegadores envolvidos na conversa, basicamente, lidam bem com tudo isso.

Se você tem alguma experiência com VOIP e conexões de vídeo, você sabe que VOIP geralmente envolve questões de proxy e firewall, bem como codecs e protocolos de sinalização, que precisam ser acordados por todas as partes envolvidas. A ideia do WebRTC é que o HTTP e a Web resolvem bem o problema de como obter dados a partir de um ponto a outro. A Web funciona. Portanto, se você tiver um navegador compatível com o padrão WebRTC (por exemplo, [Chrome e Firefox](http://www.cio.com/article/728220/Google_and_Mozilla_Hail_WebRTC_Interop_Between_Chrome_and_Firefox" \t "_blank) ), você pode usar esse navegador para se comunicar com qualquer outro cliente WebRTC.

Se alguém tem um navegador  Web com suporte WebRTC em um desktop, um smartphone ou um [dispositivo de comunicação como super-incrível relógio de pulso do futuro](http://www.cio.com/article/733977/iWatch_Sighting_in_Latest_Apple_Ad_) , você pode conversar com a pessoa em tempo real tão facilmente, e sem problemas, como se estivesse usando um telefone fixo.

**Como funciona o WebRTC?**

Em 2010, o [Google adquiriu a Global IP Solutions (GIPS)](http://www.computerworld.com/s/article/9176906/Google_to_acquire_voice_and_video_codec_company) , que desenvolveu codecs e software de comunicação em tempo real. Em 2011,  lançou o Google Hangouts, que usa a tecnologia da GIPS, e [as tecnologias de código aberto GIPS](http://www.cio.com/article/683268/Google_Releases_Video_Chat_Source_Code) na forma de WebRTC.

Atualmente, o WebRTC 1.0 é um rascunho de especificação W3C. Embora o Projeto de Trabalho já tenha sido implementado em vários navegadores, a especificação permanece em desenvolvimento.

O primeiro passo para estabelecer uma conexão de voz e de vídeo entre pares é o acesso ao microfone e à câmara de cada dispositivo. Até recentemente, estas tarefas não eram possíveis apenas com browsers. O W3C desenvolveu uma API simples chamada de [Media Capture](http://www.w3.org/TR/html-media-capture/) que ganhou algum apoio entre os fabricantes de navegadores e recentemente foi parcialmente incluída na versão móvel do Safari.

No entanto, o Media Capture não fornece qualquer meio para streaming de vídeo ou áudio. É aí que a [API MediaStream](https://developer.mozilla.org/en-US/docs/WebRTC/MediaStream_API) entra.

O trabalho da API MediaStream é perguntar ao usuário se dá permissão para acessar a câmera e o microfone e, em seguida, criar um  fluxo de áudio e vídeo sincronizado. Ele faz isso com um método JavaScript chamado getUserMedia ().

O código de base para a criar um fluxo e exibi-lo usando uma tag de vídeo HTML5 é feito modificando ligeiramente o [navigator.getUserMedia](https://developer.mozilla.org/en-US/docs/WebRTC/navigator.getUserMedia" \t "_blank), da Mozilla.org. Confira.

**< script > navigator.getMedia = ( navigator.getUserMedia ||  
navigator.webkitGetUserMedia ||  
navigator.mozGetUserMedia ||  
navigator.msGetUserMedia);  
navigator.getMedia (  
// constraints  
{  
video: true,  
audio: true  
},  
// successCallback  
function(localMediaStream) {  
var video = document.getElementsByTagName('video')[0];  
video.src = window.URL.createObjectURL(localMediaStream);  
video.onloadedmetadata = function(e) {  
// Do something with the video here.  
};  
},  
// errorCallback  
function(err) {  
console.log("The following error occurred: " + err);  
}  
);  
< /script >  
< video autoplay >< /video >**

**Mais de um forma de estabelecer uma conexão**

Uma vez que o fluxo de mídia é criado, o WebRTC usa a [API RTCPeerConnection](http://dev.w3.org/2011/webrtc/editor/webrtc.html#rtcpeerconnection-interface) para comunicar o fluxo de dados entre pares. A RTCPeerConnection, como MediaStream, emprega uma interface muito simples. É preciso um fluxo de mídia enviado para outro destinatário, onde é carregado em um RTCPeerConnection.

Sob o capô, porém, o RTCPeerConnection tem de lidar com as seguintes tarefas:

* -> Processamento de sinal (incluindo cancelamento de eco e redução de ruído)
* -> Seleção Codec
* -> Comunicação peer-to-peer
* -> Criptografia
* -> Gerenciamento de banda

Antes da transmissão entre pares começar, no entanto, um processo conhecido como sinalização deve ocorrer. Em vez de reinventar a roda, o desenvolvedor é livre para escolher qualquer protocolo de comunicação de duas vias, seja SIP, XMPP, WebSocket ou mesmo apenas JSON.

Na maioria das vezes, tudo o que é necessário para iniciar uma conexão peer-to-peer é um endereço IP público para o remetente. WebRTC usa uma [Session Traversal Utilities para NAT (STUN)](http://tools.ietf.org/rfc/rfc5389.txt" \t "_blank) servidor para dizer um aplicativo WebRTC que está por trás de um firewall seu endereço IP público. O aplicativo WebRTC pode então prosseguir com o estabelecimento de uma conexão peer-to-peer com outro aplicativo.

Aqui está um exemplo simples de como o  RTCPeerConnection funciona:

**< script > pc = new RTCPeerConnection(null);  
pc.onaddstream = gotRemoteStream;  
pc.addStream(localStream);  
pc.createOffer(gotOffer);  
  
function gotOffer(desc) {  
pc.setLocalDescription(desc);  
sendOffer(desc);  
}  
  
function gotAnswer(desc) {  
pc.setRemoteDescription(desc);  
}  
  
function gotRemoteStream(e) {  
attachedMediaStream(remoteVideo, e.stream);  
}  
< /script >**

No [Google I / O 2013](https://developers.google.com/events/io/) , a Google demonstrou um cliente de videochat completo escrito usando apenas 50 ou menos linhas de JavaScript.

Para troca de dados através de conexões peer-to-peer, o  WebRTC inclui a [API RTCDataChannels](http://www.thrupoint.com/2012/11/webrtc-data-channel/). Ela  utiliza as capacidades da RTCPeerConnection para que os usuários possam trocar qualquer tipo de dados, sem ter de se preocupar com firewall, proxy, restrições de tamanho, aplicativo de terceiros e outras dores de cabeça que têm assolado a aparentemente simples tarefa de transferir um arquivo para outra pessoa.

A RTCDataChannel utiliza a mesma API da [WebSockets](http://www.websocket.org/" \t "_blank) (tecnologia que permite a comunicação bidirecional por canais full-duplex sobre um único soquete Transmission Control Protocol, o TCP). Um método simples e o envido do manipulador de eventos OnMessage são tudo que é necessário para permitir conexões de dados bidireacionais de baixa latência. As aplicações possíveis incluem jogos, compartilhamento de tela, e compartilhamento seguro de arquivos, até mesmo de grande porte.

A RTCDataChannel também pode fornecer dados em modos confiáveis ​​ou não confiáveis. Se você precisa de dados com a menor latência possível, e um pacote possa se perder de vez em quando, o modo não confiável é o caminho a percorrer. Se os dados devem ser corretos (como na transferência de arquivos, por exemplo), use o modo confiável em seu lugar. [ShareFest](https://github.com/Peer5/ShareFest" \t "_blank) ,uma solução aberta de compartilhamento, usa RTCDataConnection para permitir o compartilhamento de arquivos de até 1 GB, sem passar por um servidor.

**WebRTC tem um potencial incrível, mas também tem limitações**

Com WebRTC, existe o potencial para resolver o maior obstáculo nas comunicações tornar a comunicação peer-to-peer de voz e vídeo (e a partilha de dados, para que o assunto) tão fácil quanto digitar mensagens através da Web hoje.

O WebRTC é suportado e ativado por padrão no Google Chrome, Chrome para Android e a última versão beta do Firefox. Pode ser usado no Internet Explorer com o [Chrome Frame](http://www.google.com/chromeframe" \t "_blank) , um plugin que permite que as tecnologias da Web aberta rodem no navegador da Microsoft. Embora o Safari ainda não suporte WebRTC, o Ericsson Labs criou um [Bowser](https://labs.ericsson.com/apps/bowser" \t "_blank) WebRTC para dispositivos iOS e Android.

Claro, esta é a Web, e nada é absolutamente fácil. Tanto a Microsoft como a Apple têm grandes investimentos em suas próprias soluções RTC. A Microsoft levantou objeções ao Google VP8 Codec e não tem suporte para WebRTC para o Internet Explorer. A posição da Apple em WebRTC não é conhecida, mas especula-se que a Apple vê WebRTC como uma ameaça para o FaceTime,, o que explica o fato de não estar em muita pressa para implementá-lo.

Depois, há o problema de como implementar as chamadas de conferência em uma rede peer-to-peer. Grupos maiores de cinco usuários apresentam problemas reais para o WebRTC, devido à complexidade da atualização e roteamento de dados de cada um dos pares para todos os outros pares.

Segurança, por sua vez, é levada a sério pelo WebRTC. Em primeiro lugar, todo acesso às câmeras e aos microfones é explicitamente opt-in. Ou seja, o navegador irá pedir ao usuário para acessar a câmera e microfone e o usuário deverá clicar no botão OK. Em seguida, todos os dados compartilhados entre os pares é criptografado usando [criptografia AES](http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf). Por último, pelo fato do WebRTC não usar qualquer plug-ins, é executado dentro do sandbox do navegador e tem apenas o mesmo acesso ao computador do usuário que uma aplicação Web tem.

**Como o WebRTC está sendo usado hoje**

WebRTC ainda está em sua infância, mas após o lançamento do [Firefox 22](http://www.engadget.com/2013/05/17/firefox-22-beta/) e versões posteriores, ele estará disponível em cerca de um bilhão de aparelhos. Dois grandes exemplos de uso WebRTC em larga escala são a [Crunched](http://www.crunched.com/" \t "_blank), uma ferramenta que permite a realização de reuniões instantâneas no navegador, e a ferramenta de colaboração de vídeo [TenHands](https://www.tenhands.net/Home.htm" \t "_blank).

Ferramentas de desenvolvimento, por sua vez, incluem o [PhonoSDK](http://phono.com/" \t "_blank), do Voxeo Labs, que pode ser usado para construir aplicativos de mensagens de voz, e os [WebRTC](http://www.webrtc.org/reference/webrtc-internals" \t "_blank)[Internals](http://www.webrtc.org/reference/webrtc-internals) para o Google Chrome.