# COMUNICAÇÃO EM TEMPO REAL ENTRE WEB BROWSERS

*João Azevedo*
*70614*

*Fábio Martins*
*71073*

Instituto Superior Técnico
Av. Prof. Doutor Cavaco Silva, 2744-016 Porto Salvo, Portugal
E-mail: {joao.ramos.de.azevedo, fabiomartins}@tecnico.ulisboa.pt

## ABSTRACT

A web é cada vez mais uma ferramenta fundamental no nosso dia-a-dia. Lemos notícias, enviamos e recebemos emails, fazemos compras, tudo de forma simples e intuitiva via um conjunto de rápidos cliques.
A plataforma que nos permite aceder a estes conteúdos, quer nos nossos computadores pessoais, quer nos nossos *tablets* e *smartphones* denomina-se de web browser. A sua capacidade de se tornar invisível ao utilizador é cada vez mais importante e por isso todas as funcionalidades que naturalmente encontramos no nosso computador devem estar disponíveis neste.
Este documento faz assim uma análise do paradigma comunicação em tempo real na web e a sua consequente implementação em web browsers.

***Index Terms***— Tempo real, web browsers, WebRTC

## 1. INTRODUÇÃO

Os últimos anos têm sido prósperos no desenvolvimento de novas soluções para, por um lado, lidar com comunicações multimédia na web; e por outro, providenciar interação direta entre diferentes utilizadores, vulgarmente conhecida por interação *peer-to-peer*. O desenvolvimento de uma solução que seja capaz de integrar estas áreas obriga a uma reformulação da pilha protocolar de aplicações, como os *web browsers* através da definição de um conjunto de protocolos e *codecs* a implementar e, todo o modo de exposição destas capacidades para os programadores de aplicações, garantindo uma fácil integração. Torna-se ainda necessário dotar a solução com a capacidade de lidar com diferentes circunstâncias como o atravessamento de NATs e Firewalls, sinalização e disposição dos *peers*.

As soluções proprietárias continuam a dominar este mercado, destacando-se o Skype e o Google Hangouts. Todavia, nos últimos tempos, temos assistido a esforços por parte da indústria na procura de uma solução normativa que confira interoperabilidade entre aplicações e, nada melhor, do que utilizar uma plataforma comum, como o *web browser*, para implementação e divulgação desta nova norma. Surgem duas propostas: WebRTC [7] e ORTC [17], e uma terceira, OpenWebRTC da Ericsson, com contornos ligeiramente proprietários.

O presente documento pretende introduzir o leitor neste ambiente, apresentando a arquitetura clássica de um sistema deste tipo, tentando ao máximo interligar com conceitos estudados na disciplina de Comunicação Multimédia. Tentar-se-á igualmente, dentro do possível, construir um documento atualizado, enquadrando as várias discussões, guerras entre vendedores, aplicações de voz e vídeo e até outras aplicações derivadas da introdução deste novo paradigma.

### 1.1. Contexto

"Um conjunto de charlatões que promovem falsas invenções para ganhos financeiros ou de autopromoção", foi como Alexander Graham Bell descreveu os alegados inventores de um sistema que juntava ao por si desenvolvido telefone a capacidade de vídeo, na primeira década do século XX. Paralelamente, nos Bell Labs da AT&T, começavam a surgir os primeiros esforços de desenvolvimento de uma tecnologia, que ao telefone juntava um televisor, com frutos visíveis comercialmente (apenas) em 1964 com o AT&T Picturephone.

Mas a verdadeira explosão da comunicação voz-vídeo estava guardada para a última década do mesmo século onde, aliando à crescente dispersão dos computadores pessoais, ao surgimento da Internet e das *webcams*, aquilo que apenas era possível na imaginação se tornou numa realidade acessível a todos nós.

Assim, em 1995 é lançado comercialmente o CU-SeeMe, uma aplicação capaz de fazer chamadas com a capacidade de vídeo adicionada à voz, ponto-a-ponto ou num ambiente de conferência recorrendo a um servidor "refletor", mais tarde chamado *conference server* ou *Multipoint Control Unit* (MCU). Os anos seguintes conduziram ao desenvolvimento de novas soluções, acompanhando as necessidades do contexto empresarial, chegando ao nosso dia-à-dia graças ao aparecimento dos sistemas gratuitos liderados pelo Skype.

Por outro caminho nascia a World Wide Web, ou vulgarmente designada por web, que, recorrendo à Internet, oferecia acesso às mais diversas páginas com conteúdos como texto ou mais ricos do ponto de vista visual como

figuras, sons e vídeos. A apresentação destes conteúdos no nosso computador recorria a uma ferramenta designada de web browser.

Em 2008 a Google, companhia que até então oferecia o motor de busca mais utilizado do mundo, decide desenvolver o seu próprio web browser. Este, naturalmente, deveria ser capaz de interpretar os conteúdos que até então se encontravam na web mas, numa tentativa de inovar, adicionar novas funcionalidades e ferramentas de interação com os mesmos. Um acompanhamento próximo e consequente implementação das normas HTML, CSS e JavaScript/ECMAScript aliado a uma interface minimalista tornou-o rapidamente no web browser mais utilizado a nível mundial.

### 1.2. Motivação

Na presença de sistemas de vídeo telefonia cada vez mais sofisticados e de uma plataforma universal a todos os computadores pessoais (hoje em dia também presente em *tablets* e *smartphones*), com capacidade de aceder a recursos na Internet, torna-se de especial interesse para os prestadores de serviços tentar fazer uso deste potencial e agregar de uma forma simples estes dois mundos. A capacidade de oferecer, por exemplo, uma chamada de vídeo e voz com um operador num *call-center* de um qualquer prestador de serviços apenas premindo um botão "Contacte-nos" presente no *website* dessa companhia torna-se uma mais-valia para lidar com situações particulares ou simplesmente de uma forma mais próxima do cliente.

### 1.3. Formulação do Problema/ Solução

No ponto histórico em que nos colocamos, encontramos um mundo onde existem sistemas de comunicação em tempo real, com partilha de áudio e vídeo associada; existem web browsers, capazes de aceder a recursos recorrendo essencialmente ao protocolo HTTP e ao estabelecimento de sessões TCP com servidores que criam a web que todos conhecemos. Mas, em que medida é que a integração destes dois sistemas é algo realmente trivial?

Na verdade, são bastantes as diferenças entre a pilha protocolar que encontrávamos nos web browsers e aquela necessária para desenvolver um sistema em tempo real.

Quando se pretende desenvolver um sistema de comunicação em tempo real com um compromisso de garantir baixa latência, evitandos atrasos, um conjunto de objetivos devem ser tidos em consideração [3,11]:

- Adaptabilidade perante a alteração das condições da rede ou do trafego nessa mesma;
- Manter um bom desempenho em redes de grande dimensão com um elevado número de ligações;
- Utilização eficiente da largura de banda disponível e baixo *jitter*;
- Ter um baixo *overhead* de bits no cabeçalho de cada pacote e ao mesmo tempo tentar garantir um baixo tempo de processamento nos diversos routers da rede mas também nas extremidades;
- Capacidade de integrar aplicações em tempo real com outras que não têm esses requisitos mas já existem na web.

Acrescendo a estes requisitos, existe ainda uma forte intenção de desenvolver uma solução normalizada para que diferentes sistemas interajam facilmente e sem terem de ser desenvolvidos *plugins* ou outros tipos de extensões.

## 2. ARQUITECTURA

Uma solução capaz de juntar a esta nova e mais sofisticada web os sistemas em tempo real, onde aplicações são embebidas nos browsers, começa por considerar a existência de duas entidades, vulgarmente conhecidas por *peers* ou pares.

A web até então encontrava-se desenhada para que quando dois *peers* pretendessem comunicar um com o outro, as suas mensagens passassem por um servidor web. Recorrendo a um exemplo clássico, quando a Alice queria enviar uma mensagem ao Bob, estabelecia uma ligação com um determinado servidor, o Bob por sua vez também estabelecia uma ligação idêntica e a mensagem era enviada primeiro para o servidor e posteriormente do servidor para o Bob [10]. O exemplo apresentado é facilmente compreendido por todos nós dado que o comportamento do servidor se assemelha em muito ao de um carteiro. Porém, facilmente conseguimos compreender que a este processo está associado um maior atraso do que quando decidimos entregar uma carta pessoalmente a um determinado individuo.

A figura 1 ilustra assim o cenário em que a comunicação é direta entre os dois indivíduos mas, acompanhando a analogia, um não sabe a morada do outro e por isso uma comunicação inicial por outro meio deve ocorrer [1].

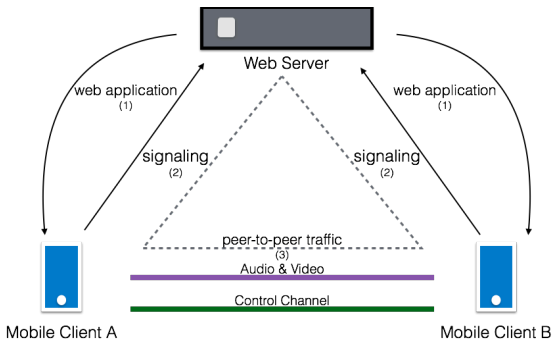


**Figura 1 – Arquitetura**

### 2.1. Protocolo UDP

No contexto web, a comunicação entre clientes e servidores recorre, usualmente, ao *Transmission Control Protocol* (TCP) [15]. Este protocolo estabelece uma sessão entre um cliente e um servidor à qual estão associadas um conjunto de

garantias, como por exemplo, na entrega de mensagens ou ordem com que são entregues. Consequentemente, os respetivos cabeçalhos dos pacotes TCP necessitam de apresentar uma maior dimensão de modo a carregar esta informação.

Surge como solução viável o *User Datagram Protocol* (UDP) [16], um protocolo de transporte, descrito no *Request for Comments* (RFC) *768* como um protocolo simples, sem estado e sem garantias, onde os seus pacotes (datagramas) carregam nada mais do que a informação necessária para a sua entrega.

A ausência de garantias providenciadas pelo protocolo UDP apresenta-se assim como uma mais-valia na redução da latência, definindo uma base sólida para a construção de sistemas de partilha de voz e vídeo.

A sua presença nos web browsers sem recorrer a nenhum *plugin* não era até recentemente uma realidade.

## 2.2. Protocolo ICE

Uma das principais diferenças entre o protocolo UDP e o protocolo TCP é a noção de sessão. No TCP quando duas entidades iniciam uma conversação uma sessão é criada e fica ativa até ser encerrada. No UDP este conceito não existe. Deste modo é esperado que ambos os intervenientes da comunicação reservem um dado porto e durante um determinado período de tempo ambos estão atentos ao trafego recebido nesse porto.

Introduza-se agora elementos novos na rede capazes de oferecer serviços de *Network Address Translation* (NAT) [20] ou firewall. Perante tais topologias de rede a comunicação direta entre os *peers* terá obstáculos maiores.

Nasce assim o protocolo *Interactive Connectivity Establishment* (ICE) [23]. Este baseia-se na criação de um agente ICE que tem como objetivo avaliar diferentes pares candidatos <endereço ip, porto> de modo a poder providenciar ao outro *peer* um endereço público no qual estará acessível. Primeiro este Agente ICE identifica, recorrendo às suas interfaces de rede, o seu endereço local, criando um par candidato com este endereço e um dado porto. Quando se encontra por trás de uma NAT e tem acessível um servidor STUN (*Session Traversal Utilities* for NAT) [19], o Agente ICE deve contactar este servidor de modo a descobrir um par <ip, porto público atribuído a si pelo NAT>. Por fim, como por vezes este procedimento não é suficiente para atravessar firewalls [12] ou *symmetric* NATs, por exemplo, recorre-se a uma ligação a servidores TURN (*Traversal Using Relays* around NAT) [14] capaz de fazer o *relay* do trafego para o outro *peer*. Temos assim um terceiro par candidato <ip do servidor TURN, porto atribuído por este>. Neste cenário a ligação entre o *peer* e o seu servidor de *relay* poderá ser TCP.

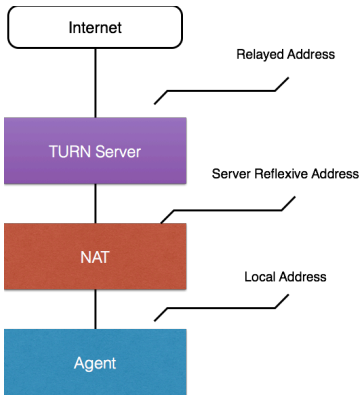


**Figura 2 - Diferentes Candidatos ICE**

## 2.3. Protocolos orientados à partilha multimédia

Não sendo o UDP um protocolo orientado à partilha de *streams* de multimédia, capaz de ter em consideração fatores como a sincronização ou a monitorização do tráfego, de modo a garantir uma determinada Qualidade de Serviço, ou mesmo a segurança da mesma comunicação, torna-se fundamental escolher/definir o protocolo indicado [2]. Descreveremos brevemente três: *Secure Real-Time Transport Protocol* (SRTP) [5]; *Stream Control Transmission Protocol* (SCTP) [21]; e *Secure Real-Time Media Flow Protocol* (SRTMFP).

O protocolo SRTP surge como o protocolo mais "leve" destes três. Caracteriza-se por incluir nos seus cabeçalhos marcas temporais (usadas para sincronização), números de sequência (para determinar perda de pacotes e reordenação) e informação sobre o tipo de dados a ser transportado (codec). A adicionar a estes campos está uma consideração com a segurança dos dados transportados recorrendo ao protocolo Advanced Encryption Standard (AES).

De seguida temos o SCTP. Este protocolo apresenta-se como um protocolo híbrido para a partilha de dados que não sejam nem de áudio nem de vídeo, num contexto de comunicação em tempo real. Este protocolo funciona assim como um misto de TCP com UDP ou melhor, um protocolo TCP "regulável", onde as garantias oferecidas pelo protocolo TCP que o distinguem do protocolo UDP podem ser incluídas ou não conforme a vontade do utilizador. Deste modo é possível, por exemplo, ter determinadas garantias na ordem das mensagens mas excluir garantias na entrega das mesmas.

Por fim, o protocolo SRTMFP foi desenvolvido pela Adobe e, aquando do projeto Cirrus, utilizado para criar redes *peer-to-peer*, utilizando a plataforma Adobe Flash, e gerir o transporte de dados multimédia. SRTMFP pode ser considerado como um pacote que não envolve só o transporte em circunstância idênticas às providenciadas pelo SRTP mas também era responsável por todo o processo de atravessamento de NATs e firewalls (idêntico ao protocolo ICE) e sinalização da chamada (estabelecimento da ligação

entre os dois *peers* recorrendo normalmente a uma terceira entidade, servidor). O facto de apenas ter sido proposto a RFC em 2013 faz com que esteja apenas associado à plataforma Adobe Flash é às soluções sobre esta desenvolvidas.

### 2.4. Codecs

A definição dos codecs multimédia a implementar nos web browsers foi um passo importante rumo à garantia de ter sistemas interoperáveis.

Assim, após duras negociações, em Novembro de 2014, o IETF definiu quais os codecs de voz/ áudio e vídeo Mandatory to Implement (MTI).

Para voz e áudio os codecs G.711 e Opus [13, 22], respetivamente e para vídeo o H.264 e VP8 [4](que devem ser ambos implementados pelos web browsers e as outras aplicações nativas podem escolher apenas um) [18].

### 2.5. Plugin? (S/N)

As primeiras implementações de sistemas de comunicação multimédia no browser faziam uso do protocolo SRTMFP, anteriormente referido, e corriam numa plataforma desenvolvida pela Adobe e denominada Adobe Flash. Estas recorriam à existência de blocos (applets), com capacidade de reproduzir conteúdo multimédia e estabelecer uma ligação, que eram embebidos na webpage. Este processo requeria por parte do utilizador a instalação de (pelo menos) o plugin do Adobe Flash. Este tipo de implementação sempre levantou algumas questões na comunidade de programadores web pelas questões de segurança associadas à execução de um applet, por um lado, e por outro da instalação do próprio plugin.

Todavia dois plugins mereceram elevado número de downloads pela grande utilização das aplicações por eles acessíveis: Skype e Google Hangouts.

A ausência de uma solução nativa ao web browser fez com que em Maio de 2011 a Google torna-se público um projeto *open-source* que envolvia a definição de um conjunto de interfaces normativas que exponham as funcionalidades oferecidas pelos protocolos orientados à partilha multimédia, nomeadamente SRTP e SCTP. O nome dado a este projeto foi WebRTC e atualmente a definição das suas interfaces está a ser feita ao nível do *World Wide Web Consortium* (W3C) e dos protocolos pelo IETF (*Internet Engineering Task Force*).

De um modo geral a implementação do WebRTC recorre a existência de um objeto JavaScript que instancia uma classe de PeerConnection com capacidade para fazer gestão de streams de áudio e vídeo (MediaStreams) [8,9] e dados (RTCDataChannel), bem como das ligações a diferentes *peers* e utilização do protocolo ICE. Estas funcionalidades estão disponíveis no Google Chrome e no Mozilla Firefox.

A porta aberta pela Google, fez com que outras soluções começassem a surgir, com arquiteturas bastante idênticas. Como empresa cotada no mercado das telecomunicações, a Ericsson tomou a liberdade de desenvolver o seu próprio browser para dispositivos iOS, o Bowser, com uma implementação idêntica ao WebRTC, denominada de OpenWebRTC, garantindo interoperabilidade com implementações dos WebRTC no Chrome e no Firefox, uma arquitetura "flexível e modular" e baseado em GStreamer.

Por fim a Microsoft, dado o seu interesse em manter viva a aplicação Skype, acabou por chegar tardiamente. De modo a não ter de implementar uma solução desenvolvida pela Google, rapidamente procurou alternativa e encontrou no projeto comunitário do W3C, Object Real-Time Communications (ORTC) uma implementação embora idêntica do ponto de vista protocolar, diferente na forma como esses mesmos protocolos são expostos. A API desta solução expõe cada um dos protocolos individualmente, permitindo ao programador web uma maior flexibilidade. É aguardada uma implementação desta solução no novo web browser da Microsoft, Microsoft Edge, cuja data de lançamento será 30 de Março de 2015.

## 3. APLICAÇÕES

Uma arquitetura como a apresentada anteriormente abre a porta ao desenvolvimento de novas aplicações por parte dos programadores web. Graças a esta torna-se possível combinar soluções em tempo real com aquelas que já encontrávamos até aqui na web.

O primeiro e natural foco do desenvolvimento de aplicações utilizando a tecnologia apresentada coincidiu com a vídeo telefonia. O desenvolvimento das versões web de aplicações como o Skype foi um dos primeiros passos para a criação de um ambiente de negócio associado a estas tecnologias. De seguida assistiu-se a um esforço concorrencial por parte de outras companhias como a Google e a Facebook mas também, ao surgimento de novas companhias que fazendo uso das funcionalidades apresentadas construíram proposta de valor para os seus clientes.

### 5.1. Skype

Revolucionário ao integrar comunicações multimédia com redes *peer-to-peer*, o Skype foi desenvolvido em 2003, e destacou-se pela facilidade de utilização, sendo difícil falar de sistemas de vídeo telefonia sem o referir.

Tendo-se tornado popular na sua versão desktop, conta com a possibilidade de chamadas de grupo com 25 pessoas em simultâneo e mediante a subscrição de determinados planos, torna-se possível realizar chamadas para telefones fixos e móveis. Os preços ao minuto dessas chamadas ronda os 1.5/4 cêntimos por minuto para a rede nacional.

Comprada pela Microsoft em 2011 num negócio que rondou os 8.5 biliões de dólares americanos, conta atualmente com cerca de 300 milhões de utilizadores com uma média de cerca de 5 milhões desses utilizadores a usarem o sistema diariamente. Estima-se que desde a sua implementação, tenham sido gastos cerca de 1.5 biliões de minutos em chamadas de vídeo e voz.

O protocolo usado para toda a comunicação é proprietário, denominando-se mesmo por "Protocolo Skype". Todavia este protocolo acaba por ser idêntico aos descritos anteriormente, fazendo uso de um tipo particular de *peers* para lidar com o problema clássico do atravessamento de NATs.

De forma a tentar abranger um maior universo de utilizadores existe uma versão para empresas capaz de reunir 250 pessoas numa só conversa, gerir contas de funcionários de forma segura e agendar reuniões através do Outlook. Esta versão está disponível por 1.5€ mensais por utilizador.

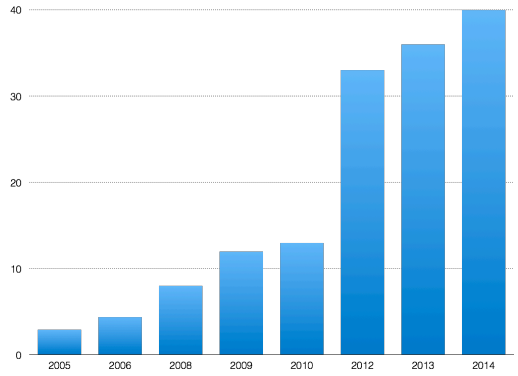


**Figura 3 - Quota de Mercado do Skype**

### 5.2. Google Hangouts

Devido ao emergente aparecimento de serviços nesta área como o Facebook Messenger, o iMessage ou o Whatsapp a gigante americana Google, viu-se obrigada a reagir com um sistema de mensagens instantâneas capaz de competir, acabando por deixar para trás o já existente serviço para o mesmo propósito da empresa, o Talk.

Quando foi introduzido em 2013, a Google prometeu que o Hangouts iria ser um dos serviços de chat mais fáceis de utilizar. A necessidade de ter apenas uma conta Google para usufruir do serviço foi vista como um ponto a favor deste sistema.

O Hangouts está disponível em versões nativas para *smartphone* ou *tablet* e ao nível do web browser. Inicialmente era fundamental a instalação de um plugin mas, hoje em dia, graças à adopção de uma solução baseada no WebRTC, esta apenas é necessária no Internet Explorer e no Apple Safari.

É adicionada à possibilidade de efetuar vídeo chamadas entre duas pessoas, uma solução para criar conversas de grupo, partilhar a área de trabalho pessoal ou mesmo a totalidade do ecrã e assim usar esta ferramenta num contexto mais profissional. Esta utilização profissional é integrada nas soluções oferecidas pela Google às empresas, pacotes a que acrescenta contas de correio eletrónico, capacidade de armazenamento e edição de documentos. O custo mínimo são cinco dólares por mês por cada utilizador empresarial ou cinquenta dólares anuais.

### 5.3. Facebook Video Chat

Em 2011, a Facebook decidiu aliar à maior rede social a nível mundial uma forma de os utilizadores comunicarem com os seus amigos, nascendo o Facebook Video Chat.

Para a sua utilização, os clientes necessitavam numa primeira fase de instalar o plugin do Skype, algo que acabou por ser descontinuado em prol de uma solução que recorre ao WebRTC tornando a comunicação áudio-vídeo apenas disponível no Chrome e Firefox.

O facto deste serviço estar incorporado no próprio Facebook, sendo quase como um *add-on* ao mesmo, faz com que a capacidade de gestão de amigos e pessoas com quem desejamos falar seja feita de forma simplificada.

A ausência de soluções viradas para o mercado empresarial faz com que este produto continue de utilização estritamente gratuita.

### 5.4. PeerCDN

A dado ponto a palavra comunicação associada a este tipo de sistemas começou a ser trocada por partilha, fazendo-nos mergulhar num oceano de soluções que não têm por base a vídeo telefonia ou videoconferência mas sim o paradigma *peer-to-peer*.

Surge assim o PeerCDN, uma solução de entrega de conteúdos *peer-to-peer* com o objetivo de reduzir o uso da largura de banda no servidor. O conceito de *Content Delivery Network* (CDN) não é novo e assenta na existência de servidores, normalmente mais próximos do cliente, com os mesmos conteúdos que o servidor web da página a que o cliente está a aceder. Desta forma, em vez de sobrecarregar o servidor, o tráfego dos clientes é distribuído. A empresa líder neste ramo é a Akamai e tem cerca de 170.000 servidores em mais de 100 países, capazes de servir estes propósitos [6].

Sendo assim, o principal propósito do PeerCDN é de dotar individualmente cada utilizador ativo num determinado instante, num determinado website da capacidade de partilhar uma imagem que acabou de receber do servidor com novos utilizadores do mesmo website. Através da rede criada, conseguida sobre a API WebRTC DataChannel, a largura de banda utilizada por parte do servidor aquando da troca de dados entre os *peers* é nula, diminuindo assim os custos do sistema.

Em 2013 a Yahoo adquiriu o PeerCDN por uma quantia não declarada publicamente.

### 5.5. BananaBread

BananaBread é um jogo 3D multiplayer online do tipo “first person shooter” que corre na web. Recorrendo a tecnologias como o WebGL e o WebRTC é capaz de introduzir a realidade dos videojogos no web browser, de forma simples e sem plugins.

Segundo os próprios produtores do jogo, este é visto como um teste para demonstração do potencial atual dos sistemas web, avaliando o seu desempenho e permitindo apontar melhorias a implementar. Ao mesmo tempo existe uma forte vontade de desmistificar a capacidade de desenvolver videojogos recorrendo à linguagem JavaScript.

Por fim um detalhe que aproxima este projeto da filosofia do WebRTC, todo o seu código fonte se encontra disponível publicamente de modo a poder servir de base para novos projetos.

### 5.6. Partilha de ficheiros

Quando se fala em *peer-to-peer*, instantaneamente as pessoas associam a partilha de ficheiros e downloads (ilegais).

Recentemente, o mesmo criador do PeerCDN apresentou ao mundo o WebTorrent, um cliente de BitTorrent com a capacidade de ser desenvolvido em JavaScript. Desta forma o WebTorrent é capaz de se executar quer no web browser, quer em servidores node.js. Numa fase inicial apenas era possível comunicar com clientes WebTorrent mas hoje em dia já é possível efetuar partilha de dados com qualquer cliente BitTorrent. Mais uma vez, este sistema recorre ao WebRTC, mais concretamente à API RTCDataChannel presente nos web browsers Google Chrome e Mozilla Firefox.

## 4. FUTURO

Conforme apresentado anteriormente, a busca de uma solução normativa tem, como sempre nestas situações, originado disputas pela sobreposição de interesses. Considerando três dos principais *players* do mercado: Google, Microsoft e Apple, o consenso não tem sido fácil de alcançar. Vejamos, quando a Google propõe uma solução as outras duas demoram a responder, até que se começam a envolver no processo e apenas muito tempo depois buscam fazer parte da solução ou desenvolver a sua "própria solução normativa". A Apple, aqui referida, acaba por ser praticamente excluída da totalidade do documento uma vez que se tem autoexcluído da discussão. O principal motivo para esta atitude é a existência da aplicação nativa, principalmente vocacionada para o Universo Apple, disponível tanto em MacOS como nas plataformas móveis iOS, o FaceTime.

Por outro lado existe o mundo das operadoras de telecomunicações. Estas, sentido o foco do seu negócio a migrar da voz para os dados móveis, temem tornar-se gestores de uma mera rede de transporte de dados, tal como existe por exemplo no mercado da Energia. O aparecimento de soluções desenvolvidas pelo Skype, Google, Facebook, etc., vulgarmente designadas de Over-the-Top (OTT) fez com que algum valor acrescentado fosse-se dissipando.

Peculiarmente, as operadoras de telecomunicações acabam por procurar no WebRTC e noutros esforços de normalização uma nova afirmação, uma tentativa de antecipação e consequente tentativa de ganhar terreno para as ditas OTTs. Por exemple, o projeto europeu reThink, que tem por objetivos desenhar as redes de comunicação do futuro, propõe a criação de várias instâncias de servidores TURN na rede, capazes de direcionar tráfego especifico e desse modo cobrar um novo conjunto de serviços.

Por fim existe o problema do *peer-to-peer*. Dado o seu historial de uso para fins menos lícitos, existe uma grande apreensão por parte dos produtores de conteúdos quando surge uma solução deste tipo. Surge assim a necessidade de prova do real valor de uma solução como a apresentada, mesmo que por vezes o seu uso possa não ser o mais indicado.

## 5. CONCLUSÃO

Após uma análise dos componentes necessários para criar uma arquitetura capaz de suportar comunicações em tempo real entre web browsers, concluímos a sua enorme importância no desenvolvimento de uma web mais sofisticada e robusta onde uma plataforma, o web browser se afirma como acessível em qualquer dispositivo.

O levantamento dos requisitos dum sistema deste tipo levou-nos a conhecer três abordagens normativas desenvolvidas recentemente, acrescidas às soluções proprietárias que no passado já se encontravam disponíveis (mediante instalação de *plugins*, etc.).

Providenciando melhores ferramentas aos programadores web, tornamos possível o aparecimento de aplicações que melhor satisfazem as necessidades dos utilizadores, sejam elas para partilha de voz e vídeo ou de outro qualquer tipo de dados.

## 6. AGRADECIMENTOS

Dois agradecimentos de especial consideração: BlogGeek.me pelos artigos de especial relevância que em muito ajudaram a definir opiniões e compreender as questões envolvidas e HTML5 UP por algum código reaproveitado ao nível do CSS e JS da aplicação web (html5up.net).

## 7. REFERÊNCIAS

[1] Alvestrand,H.:Overview:Realtimeprotocolsforbrowser-basedapplicationsdraft- ietf-rtcweb-overview-13. Tech. rep., Internet-draft, IETF Network Working Group (2012)

[2] Amirante, A., Castaldi, T., Miniero, L., Romano, S.P.: On the seamless interaction between webrtc browsers and sip-based conferencing systems. Communications Magazine, IEEE 51(4), 42–47 (2013)

[3] ARAS, C¸.M., KUROSE, J.F.: Real-time communication in packet-switched net- works. PROCEEDINGS OF THE IEEE 82(1) (1994)

[4] Bankoski, J., Koleszar, J., Quillio, L., Salonen, J., Wilkins, P.: Y. xu," vp8 data format and decoding guide. Tech. rep., RFC 6386, November (2011)

[5] Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: Rfc 3711: The secure real-time transport

protocol (srtp). Dostupn´e z:¡ http://www. ietf. org/r-fc/rfc3711. txt (2004)
[6] Belson, D., Thompson, J., McKeay, M., Brenner, B., Mo¨ller, R., Sintorn, M., Hus- ton, G.: Akamai's - state of the internet q3. Tech. rep., Akamai (2014)
[7] Bergkvist, A., Burnett, D.C., Jennings, C., Narayanan, A.: Webrtc 1.0: Real-time communication between browsers. Working draft, W3C 91 (2012)
[8] Burnett, D., Narayanan, A., Bergkvist, A., Jennings, C.: Media capture and streams. World Wide Web Consortium WD WD-mediacapturestreams-20130903 (2013)
[9] Burnett, D.C., Narayanan, A.: getusermedia: Getting access to local devices that can generate multimedia streams. W3C Editor's Draft (2011)
[10] Chainho, P., Haensge, K., Druesedow, S.: Signalling-on-the-fly: Sigofly (11 2014), webRTC Interoperability tested in contradictive Deployment Scenarios
[11] Ferrari, D.: Client requirements for real-time communication services (1990)
[12] Johnston, A., Yoakum, J., Singh, K.: Taking on webrtc in an enterprise. Commu- nications Magazine, IEEE 51(4), 48–54 (2013)
[13] Marocco, E., Berger, E., Marjou, X., Jesske, R., Bogineni, K., Proust, S., Lei, M.: Webrtc audio codecs for interoperability with legacy networks. (2013)
[14] Matthews,P.,Rosenburg,J.,Mahy,R.:Theinternetengineeringtaskforcerfc5766: Turn-traversal using relays around nat (2010)
[15] Postel, J.: Rfc 793: Transmission control protocol, september 1981. Status: Stan- dard 88 (2003)
[16] Postel, J., Protocol, U.D.: Rfc 768. User datagram protocol pp. 1–3 (1980)
[17] Raymond, R., Hookflash, Aboba, B., Uberti, J.: Object rtc (ortc) api for webrtc. Draft Community Group Report (October 2014)
[18] Roach, A.: Webrtc video processing and codec requirements. draft-ietf-rtcweb- video-03 (work in progress) (2015)
[19] Rosenberg, J., Mahy, R., Matthews, P., Wing, D.: Rfc 5389: Session traversal util- ities for nat (stun). Internet Engineering Task Force (2008)
[20] Senie, D.: Network address translator (nat)-friendly application design guidelines. rfc3235. 2002 10. hao f, anderson r, daugman j. combining crypto with biometrics effectively. IEEE Transactions on Computers 55(9), 1081–1088 (2006)
[21] Stream, R.: Stream control transmission protocol. RFC 4960 (2007)
[22] Valin, J., Bran, C.: Webrtc audio codec and processing requirements. draft-ietf- rtcweb-audio-05 (work in progress) (2014)
[23] Westerlund, M., Perkins, C.: Iana registry for interactive connectivity establishment (ice) options (2011)

**João Azevedo** Aluno a frequentar o quinto ano letivo do curso de Engenharia de Telecomunicações e Informática a realizar a dissertação na integração de sensores em comunicações multimédia na web; Investigador no INESC-ID na área de segurança em software e hardware; Voluntário IEEE.

**Fábio Martins** Aluno a frequentar o quinto ano letivo do curso de Engenharia de Telecomunicações e Informática a realizar a dissertação na área do desenvolvimento de sistemas inteligentes para previsão em mercados financeiros; Membro do NEETI.