Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2013

João Luís Borges Barbosa

Desenvolvimento de uma aplicação colaborativa baseada em WebRTC

João Luís Borges Barbosa

Desenvolvimento de uma aplicação colaborativa baseada em WebRTC

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre no Mestrado em Sistemas de Informação, realizada sob a orientação científica do Prof. Dr. Diogo Nuno Pereira Gomes, Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Dr. Paulo Jorge Salvador Serra Ferreira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

No meu último esforço académico dedico esta dissertação à Joana Gomes, pelo apoio e dedicação incomparáveis.

o júri

presidente

Prof. Dr. Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais

Dr. Eduardo Rinn Estanqueiro Rocha
Investigador associado da Leipzig University Of Applied Science

Dr. Diogo Nuno Pereira Gomes
Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Agradeço desta forma aos meus orientadores Diogo Gomes e Paulo Salvador pela instrução e conhecimentos prestados.
Quero agradecer à Wavecom pela oportunidade oferecida e a todas as pessoas da mesma pelo seu convívio, destacando as equipas que constituem a Finesource.
Acrescento nesta lista a minha família e os ensinamentos que me proporcionaram, os meus amigos, colegas e pessoas conhecidas ao longo da minha vida, por contribuírem de uma maneira ou de outra para quem sou.

palavras-chave WebRTC, aplicação colaborativa *web*, HTML5, VoIP, SIP

resumo

A comunicação desenrolou um papel fundamental na evolução do ser humano. Com o advento dos telefones tornou-se possível comunicar à distância, mas apenas a voz era transmitida.

O desenvolvimento das tecnologias permitiu posteriormente a troca de vídeo entre dois pontos longínquos, mas as infra-estruturas eram limitadas.

A Internet veio oferecer a permuta de informação de forma eficiente e adaptável, características apelativas para as comunicações em tempo real. A banalização deste conjunto de tecnologias permitiu às empresas baixar os seus custos ao integrar a telefonia com esse mesmo conjunto. Esta acção tornou-se uma necessidade proveniente da crise económica instalada nos últimos anos. Nesta mudança acrescenta-se o benefício das entidades empresariais poderem desenvolver interacções intrínsecas entre os seus serviços e a telefonia.

Os aperfeiçoamentos aos conteúdos multimédia continuam actualmente a vários níveis, sejam equipamentos ou mecanismos dedicados à qualidade dos mesmos, tudo devido às implicações das comunicações em tempo-real. Uma parte interessante deste progresso é o uso da voz e vídeo em diversos ambientes colaborativos, como reuniões corporativas, jogos *online* ou actividades lúdicas. Para estes fins, a diversidade de aplicações é crescente mas ainda limitada, requerendo conhecimentos de instalação ou configuração que podem criar dificuldades de usabilidade ao utilizador típico da Internet.

Neste documento é proposta uma solução capaz de minimizar os obstáculos que as soluções actuais apresentam aos seus utilizadores. Baseada em HTML5, esta aplicação oferece um serviço onde três ou mais intervenientes têm a habilidade de comunicar e colaborar entre si, com recurso exclusivo ao seu *browser*. Será realizado um estudo das tecnologias *web* emergentes para adquirir as bases tecnológicas essenciais a serem implementadas no sistema designado.

keywords WebRTC, collaborative web application, HTML5, VoIP, SIP

abstract

Communication unrolled a key role in human evolution. With the advent of mobile communications it became possible to communicate at a distance, but only the voice was transmitted.
Later technology development allowed the exchange of video between two distant points, but the infrastructure was limited.
The Internet has to offer exchange information efficiently and adaptively, appealing features for real-time communications. The banality of this set of technologies enabled companies to lower their costs by integrating telephony for the same. This action has become a necessity installed from the economic crisis in recent years. This change builds up the benefit of the business entities that can conceive close interactions between its services and the media referred.
The improvements to multimedia content currently continue at various levels, equipment or mechanisms are dedicated to the quality of them, all due to the implications of communications in real-time. An interesting part of this progress is the application of voice and video in multiple collaborative environments, such as business meetings, online games or play activities. For these purposes, the range of applications is growing but still limited, requiring knowledge of installation or configuration, creating difficulties to the typical Internet user.
In this document it's proposed a solution that would minimize the obstacles that current solutions present to its users. Based on HTML5, this application offers a service where three or more participants have the ability to communicate and collaborate requiring only their browser. A detailed study of emerging web technologies will be made to acquire the essential technological bases to be implemented on the target system.

Índice

Índice de Figuras

Índice de Tabelas

Lista de acrónimos

AEC	Acoustic Echo Canceler
AES	Advanced Encryption Standard
AIM	AOL Instant Messenger
AOL	America Online
API	Application Programming Interface
BLOB	Binary Large Object
CRM	Customer Relationship Management
CSRC	Contributing Source
CSS	Cascading Style Sheets
DOM	Document Object Model
DRY	Don’t Repeat Yourself
DTLS	Datagram Transport Layer Security
DTMF	Dual-Tone Multi-Frequency
FEC	Forward Error Correction
FIR	Full Intra Request
GI	Global Index
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
IdP	Identity Provider
IP	Internet Protocol
IETF	Internet Engineering Task Force
iLBC	Internet Low Bit Rate Codec
iSAC	Internet Speech Audio Codec
JID	Jabber ID
JSEP	JavaScript Session Establishment Protocol
MCU	Multipoint Unit Controller

MGCP	Media gateway control protocol
MIME	Multipurpose Internet Mail Extensions
MTU	Maximum Transmission Unit
NACK	Negative Acknowledgements
NAPT	Network Address and Port Translation
NAT	Network Address Translation
NR	Noise Reduction
PBX	Private Branch Exchange
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
PLI	Picture Loss Indication
PPID	Payload Protocol Identifiers
PSTN	Public Switched Telephone Network
QoS	Quality of Services
RCP	Remote Call Procedure
RTCP	RTP Control Protocol
RTMP	Real Time Messaging Protocol
RTP	Real-time Transport Protocol
RTP/AVPF	RTCP-based feedback
RTP/SAVPF	Extended Secure RTP Profile for RTCP-Based Feedback
RTSP	Real Time Streaming Protocol
SASL	Simple Authentication and Security Layer
SCTP	Stream Control Transmission Protocol
SDK	Software Development Kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLI	Slice Loss Indication
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRTP	Secure Real-time Transport Protocol
SSL	Secure Sockets Layer

SSRC	Single Synchronisation Source
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TMMBR	Temporary Maximum Media Stream Bit Rate Request
TSTR	Temporal-Spatial Trade-off Request
TURN	Traversal Using Relays around NAT
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
VoIP	Voice over Internet Protocol
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
XML	eXtensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

1. Introdução

Com a origem da Internet, o seu impacto na sociedade não parou de aumentar, tornando-se indispensável no quotidiano. Tornou-se um meio de comunicação de eleição com a capacidade de transmitir qualquer tipo de dados a elevadas velocidades e com diferentes camadas de definições que podem incorporar várias tecnologias. Tendo estas características, a transição de áudio e vídeo para a Internet foi impulsionada pelas dificuldades económicas que surgiram ao longo do tempo, constituindo assim uma oportunidade para melhorar a qualidade de todos os níveis destas formas de comunicação.

Dada a natureza da Internet, os problemas de congestionamento de tráfego podem comprometer a qualidade do áudio e vídeo de uma sessão, e a necessidade de encontrar soluções para conseguir ultrapassar estas questões foi introduzida como consequência. Para combater essas debilidades, foram criados vários mecanismos, como por exemplo, Quality of Service (QoS), indicado para resolver algumas das principais dificuldades através da diferenciação dos tipos de tráfego existentes, protocolos dedicados à transmissão de multimédia e *codecs* optimizados para redes informáticas [1]. O *hardware* teve a sua evolução disponibilizando maior largura de banda, ligações com melhor performance, processamento de dados mais rápido e aumento do tamanho da *cache*. Apesar do progresso feito em todas as áreas informáticas, a problemática de congestionamento de tráfego prevalece. Juntam-se a este facto as novas definições de imagem com elevado detalhe, gerando um aumento de dados. A exigência para encontrar novas resoluções em relação à problemática referida é constante e inerente nas tecnologias emergentes, que tentam conjugar diferentes técnicas de comunicação em tempo real com soluções de engenharia originais.

Actualmente estão disponíveis diversos *softwares* para transmitir áudio e/ou vídeo que oferecem funcionalidades adicionais para proporcionar uma experiência interactiva completa. Algumas das opções disponíveis por defeito incluem mensagens instantâneas, partilha de ficheiros, visualização de imagens e painel de desenho. Numa sessão entre duas pessoas a prestação de todas as funcionalidades consegue ser excelente na Internet actual mas o maior desafio são as conferências ou sessões colaborativas. Estas últimas têm grande

interesse empresarial, poupando custos e tempo, mas a capacidade de processamento, principalmente dos terminais, e largura de banda precisam de ser abundantes para lidar com as múltiplas *streams* associadas, ficheiros da sessão e outros dados. Ao instalar aplicações para esta finalidade estamos dependentes de quem as criou para serem suportadas em qualquer dispositivo terminal e possivelmente limitados à Internet actual. Implementando uma solução *web* completa com arquitectura cliente-servidor numa rede privada as restrições são menores, no entanto é necessário recorrer a um *browser* com os devidos *plugins* para realizar conferências, encontrando dificuldades de segurança e permissões assim como falta de suporte em alguns dispositivos. Surgiram propostas para *standards* incluídos na versão 5 do HyperText Markup Language (HTML) que oferecem a oportunidade de desenvolver novas aplicações omitindo qualquer intervenção de programas externos [2]. Com as funções resultantes dessa standardização, são esperados avanços significativos nas comunicações em tempo real por intermédio de ferramentas que integram tecnologias e processos, que no seu conjunto formam uma potencial solução para todos os problemas. Neste trabalho as componentes que constituem o HTML5 serão objecto de estudo com o desenvolvimento de uma aplicação *web* de colaboração baseada nas mesmas.

Esta dissertação foca-se nas problemáticas das aplicações colaborativas e contribui com o estudo das tecnologias *web* emergentes na tentativa de solucionar as limitações actuais. Também faz parte dessa contribuição uma solução *web* baseada em HTML5 onde três ou mais intervenientes podem comunicar e colaborar entre si com recurso exclusivo ao seu *browser*.

1.1. Objectivos

As tecnologias da nova versão do HTML dedicadas às comunicações em tempo real contêm inovações para esta área, destacando a inexistência de *software* adicional para transmitir vídeo e áudio, assim como a extensibilidade das mesmas. Chamadas entre telefones e utilizadores WebRTC são um bom exemplo desta última característica e um dos objectivos deste documento [3]. A ausência de um cliente HTML5 apelativo para o público em geral com as funcionalidades essenciais para uma experiência colaborativa é

notória, sendo necessário um servidor com suporte para coordenar as informações para que esse tipo de experiências se torne uma realidade. Com estes aspectos identificados, os pontos-chave desta dissertação são os seguintes:

- Pesquisa das aplicações colaborativas actuais
- Estudo das tecnologias HTML emergentes que implementam mecanismos para comunicações em tempo real
- Elaboração de uma arquitectura para um sistema de comunicações em tempo real
- Implementação de um sistema de colaboração *web* baseado em HTML5

É imperativo que haja um esforço para encontrar as melhores ferramentas e o contexto onde serão introduzidas com o fim de elaborar uma plataforma de apoio cooperativo.

1.2. Conteúdos

Capítulo 2 é constituído pelo estado de arte dos programas colaborativos e a sua evolução. Estão representados *softwares* como Windows Messenger, Skype, BigBlueButton, Lynckia e WebEx que apresentam diferenças em variados níveis mas contém a mesma finalidade em termos de comunicações. Conjuntamente introduz o *standard* HTML5, distinguindo as suas novidades, principalmente aquelas que são relevantes para o transporte de dados em tempo real e meios de comunicação, o seu suporte e as suas limitações. O conceito Voice over IP (VoIP) também é apresentado neste capítulo.

Capítulo 3 pormenoriza o WebRTC e as suas componentes, incluindo a sua arquitectura, as tecnologias envolvidas, as suas restrições e potencialidades. Algumas capacidades exploradas estão identicamente descritas neste ponto.

Capítulo 4 enquadra outras tecnologias de transporte de dados que podem ser integradas com WebRTC e são identificadas as suas vantagens e desvantagens face às suas capacidades para transmissões de informação em tempo real.

Capítulo 5 apresenta o desenvolvimento completo da aplicação proposta, a par com a discussão das ferramentas envolvidas na sua implementação e a validação da sua performance.

Capítulo 6 reúne as conclusões sobre o *software* desenvolvido e as tecnologias estudadas nesta dissertação, além do trabalho futuro relacionado com a aplicação colaborativa produzida.

2. Estado da arte

Quando a Internet foi disponibilizada para o mundo, começou a mudá-lo e tornou-o numa "aldeia", aproximando as pessoas, independentemente da distância que as separa. A área de comunicações começou a tirar proveito da Internet e desenvolveu aplicações com essa intenção. Actualmente, existem boas soluções para comunicar, assim como para partilhar dados entre os intervenientes. As suas funcionalidades tornam-nas bastante úteis e versáteis, sejam para chamadas privadas ou para conferências. No dia 1 de Março de 1936 foi realizada a primeira chamada com áudio e vídeo entre duas pessoas na Alemanha passando, este serviço, a estar ao dispor do público [4]. Anos depois a Internet tornou-se popular, tecnologias de áudio e vídeo evoluíram, graças à sua inclusão na Internet, foram desenvolvidas ferramentas para colaboração *online* e o próximo passo é a transmissão destes tipos de meios de comunicação pela World Wide Web, com suporte para conferências.

Os produtos para comunicar em tempo real podem ser classificados pelos seus objectivos, formando duas categorias:

- **Chamadas Privadas:** tipo de chamada que envolve dois participantes.

- **Conferências:** Chamadas com três ou mais participantes em que poderão ser importantes funcionalidades adicionais, como visualização de apresentações ou outros documentos. Apresentam maior interesse para empresas, dado que facilitam as suas interacções.

As soluções actuais para comunicações através da Internet oferecem uma variedade de funcionalidades comuns entre elas, mas são diferentes na sua arquitectura e características técnicas. Estas últimas são mais relevantes para as empresas devido aos seus requisitos, sendo menos interessante para o cliente comum que deseja realizar chamadas particulares.

Este capítulo mostra o estudo realizado acerca das distintas aplicações de modo a adquirir os conhecimentos das implementações concretizadas e as suas limitações com a finalidade de conceber uma aplicação capaz de apresentar algo inovador. Essa mesma inovação surgirá com maior enfâse nas tecnologias utilizadas do que nas capacidades da interface.

2.1. Voice over Internet Protocol

As comunicações de voz através de redes informáticas são definidas por este conceito, que envolve diversas tecnologias para tornar a telefonia numa parte da Internet. A viabilidade deste cenário reside na garantia da troca de dados em tempo real. Para tal suceder são considerados os passos para estabelecer uma chamada através da Internet e elementos como, perda de pacotes, novos *codecs*, atrasos e as suas variações, e arquitectura de rede para suportar QoS. [5]

Num sistema VoIP, a voz é digital, portanto será necessário converter a mesma caso os dispositivos envolvidos numa chamada sejam analógicos. Esta diferença entre equipamentos analógicos e digitais normalmente advém da interligação de redes VoIP e Public Switched Telephone Network (PSTN), conseguida via servidores *gateway*, encarregues da "tradução" de informações das duas redes, podendo incluir a conversão entre voz digital e voz analógica. A negociação de *codecs* de uma chamada também é realizada pelos últimos servidores mencionados, mas se não existir um *codec* em comum no estabelecimento da chamada, então será efectuada uma conversão por parte do servidor denominada *transcoding*. [5]

Em relação às componentes de transporte para sistemas VoIP, os protocolos Real-time Transport Protocol (RTP), User Datagram Protocol (UDP) e Internet Protocol (IP) são os destacados para garantir as comunicações em tempo real. Dado que as retransmissões de pacotes podem provocar atrasos, os protocolos com estas características não se adequam ao contexto VoIP. Outros factores particulares ao contexto referido que podem gerar atrasos abrangem a criação de pacotes a enviar pela rede, processamento de *codecs* e congestionamento de tráfego. Podem ser introduzidos mecanismos QoS para diminuir os atrasos encontrados numa rede informática. [5]

Para sinalizar sessões através da Internet, existem protocolos específicos para esse fim, tais como Session Initiation Protocol (SIP), H.323, Media Gateway Control Protocol (MGCP) e Megaco. Os dois primeiros aplicam o controlo da sinalização *peer-to-peer*, enquanto os restantes apoiam esse controlo do tipo *master-slave*. Adicionalmente, os protocolos H.323 e Megaco suportam videoconferências além de telefonia, e o protocolo SIP foi construído especificamente para a Internet, integrando suporte para diferentes tipos de sessões. [5]

VoIP é uma área informática cada vez mais adoptada, com tecnologias constantemente evoluídas e constitui uma das bases para esta dissertação.

2.2. Windows Live Messenger

O *software* aqui mostrado, inicialmente designado de MSN Messenger, é um cliente de mensagens instantâneas desenvolvido pela Microsoft com suporte para vários sistemas operativos, como Microsoft Windows, Symbian OS, Mac, iOS, etc. O serviço MSN Messenger, anunciado no dia 21 de Julho de 1999, dava a possibilidade de comunicar com utilizadores registados no MSN Hotmail, serviço de *web-email* da Microsoft, ou os que usufruíam do AOL Instant Messenger. Foi o primeiro serviço interoperável com outras aplicações e serviços, sendo alguns Microsoft Internet Explorer, Outlook Express e NetMeeting [6], [7]. Ao longo do tempo foram adicionadas novas capacidades ao cliente da Microsoft, mencionadas nos próximos subtópicos.

Comunicações

Na parte de comunicações, as funcionalidades incluídas foram: comunicação por voz, através de tecnologia Net2Phone para telefones convencionais e mais tarde ponto-a-ponto ou servidores MSN, assim como vídeo [8], [9]; transferência de ficheiros, com uma evolução fundamentada em Universal Plug and Play (UPnP) [10]; suporte para protocolo *msnim*, com a capacidade de adicionar contactos ou iniciar conversas a partir de endereços encontrados em páginas *web* [11]; mensagens Instantâneas Offline [12]; possibilidade de

comunicar com utilizadores de diferentes clientes de mensagens instantâneas, como AIM, Google Talk e ICQ [13]; mensagens instantâneas em sessões multi-utilizador [13].

Interface

A interface integrava as seguintes características: escrita manual [14]; ícones, fundos e imagens de perfil animados [14]; personalização de perfil [14]; jogos *online* [15].

Outras funcionalidades

As restantes funcionalidades constituem: acesso com qualquer conta de correio electrónico; notificações de correio electrónico nas contas Hotmail; integração com o Windows Media Player, Apple iTunes, Bing, Xbox Live, Hotmail e redes sociais [16]; pastas partilhadas, substituídas posteriormente pelo serviço SkyDrive [17], [18]; instância do *software* em diferentes equipamentos com a mesma conta associada [18]; importação e exportação de contactos através de ficheiros CTT [19].

Para completar e integrar o Windows Live Messenger, os programadores tinham à sua disposição algumas Application Programming Interfaces (API) inseridas em diversos Software Development Kits (SDK), e outras ferramentas com propósitos distintos no desenvolvimento de módulos [20]. Um dos SDKs, o Windows Live Messenger Activity SDK, tem como objectivo fornecer as componentes técnicas e a sua informação para explorar aplicações *web* que envolvam actividades com contactos no Windows Live Messenger, como jogos [21]. Outro conjunto de funções é o Windows Live Messenger Web Toolkit, que torna possível uma integração avançada com aplicações *web*, ao aceder a informações de contactos, controlar dados de perfil e tirar partido de comunicações em tempo real [22]. Para utilizar o mesmo conjunto de funções, existem duas opções:

- **Windows Live Messenger UI Controls:** conjunto de funções básicas baseadas em HTML para quem desenvolve aplicações interactivas com o cliente em questão. Personaliza elementos visuais e o seu comportamento [23].

- **Windows Live Messenger Library:** biblioteca JavaScript com funções avançadas para situações em que as equivalentes do ponto anterior não são suficientes [24].

A ferramenta Windows Live ID SDK é constituída por dois SDKs para diferentes cenários das aplicações criadas por terceiros [25]. São elas:

- **Windows Live ID Delegated Authentication SDK for Application Providers:** componente disponível para aplicações que interagem com serviços do Windows Live e *sites web* que requerem autenticação por intermédio do Windows Live ID, em nome do utilizador com uma conta Windows Live [26].

- **Windows Live ID Web Authentication SDK:** disponibiliza os conteúdos para realizar autenticação numa aplicação *web* independente, ao comunicar com os respectivos serviços Windows Live [27].

O Windows Live Messenger Connect é uma API que envolve outras semelhantes, como Windows Live ID e Windows Live Messenger Webkit, seguindo algumas especificações e *standards* definidos, incluindo Java, PHP: Hypertext Preprocessor (PHP) e Flash para facilitar a sua integração. Inclui funções e elementos com o objectivo de melhorar as aplicações *web* que interagem com o Windows Live Messenger. Algumas funcionalidades que a aplicação integrante da última API pode apresentar, caso um utilizador conceda autorização, abrangem o acesso a informações do utilizador e às suas fotos no SkyDrive e a comunicação através do Windows Live Messenger [28]. Por fim o Windows Live Admin Center SDK efectua ligações Simple Object Access Protocol (SOAP) Remote Call Procedure (RPC) aos servidores do Windows Live Admin Center para gerir os utilizadores de um domínio registado nesta plataforma da Microsoft, por intermédio de uma aplicação *web* externa. A estrutura desta ferramenta consiste num programa exemplo Win32 para gestão de membros de um domínio, duas aplicações exemplo executadas pela linha de comandos para o mesmo fim, uma aplicação para actualizar um ficheiro CSV com uma lista de clientes de um domínio personalizado do

Windows Live Admin Center e a documentação para as funções SOAP. As primeiras três aplicações estão codificadas em C# e os seus códigos-fonte fazem parte do SDK referido anteriormente [29].

O cliente Windows Live Messenger teve o seu fim em Abril de 2013 e a Microsoft migrou os seus clientes para a aplicação Skype [30].

2.3. Skype

O Skype é um *software* que disponibiliza comunicações através de mensagens instantâneas, de voz, de vídeo, Short Message Service (SMS) e partilha de recursos [31]. É compatível com vários sistemas operativos, como Linux, Microsoft Windows e Mac, além de ser suportado por outras plataformas, incluindo Windows Phone, Android, iOS e televisões [32].

As funcionalidades específicas do Skype revelam a versatilidade desta aplicação, que consegue comunicar com praticamente qualquer pessoa do mundo. Para tal, este *software* surge com capacidades de realizar chamadas grátis entre Skypes, chamadas para telefones ou telemóveis, com a opção de usar um número do cliente para identificar as mesmas, ou através de um *plugin* Skype pertencente a um *browser* adquirir um número de telefone para ligar a partir de qualquer dispositivo móvel com tarifas Skype. Outras capacidades deste programa resumem-se a efectuar videochamadas entre duas ou mais pessoas, enviar mensagens de voz, vídeo e texto para contactos *offline* e SMS, partilhar ficheiros e ecrã, integrar redes sociais e Private Branch Exchange (PBX) com suporte SIP, para comunicações VoIP através do Skype Manager [31], [33], [34].

A versão do Skype para dispositivos móveis também se encontra disponível, além de uma aplicação designada GroupMe. Esta última destaca-se por dar a possibilidade de formar grupos onde é possível comunicar por SMS em telemóveis convencionais e por mensagens instantâneas em *smartphones*, aumentando as funcionalidades nos segundos com partilha de localização actual e fotografias além de integração com redes sociais [35].

A arquitectura da aplicação mencionada no título deste subtópico e o seu modo de funcionamento não são totalmente conhecidos, tornando-se assim alvo de pesquisas que conseguiram demonstrar alguns dos seus detalhes. A sua arquitectura foi baseada numa

rede ponto-a-ponto, onde o processamento de dados e encaminhamento de tráfego eram efectuados pelos computadores ou nós que pertencem a essa rede, logo descentralizados e independentes de qualquer servidor. A Figura 1 demonstra a possível arquitectura parcial mencionada.

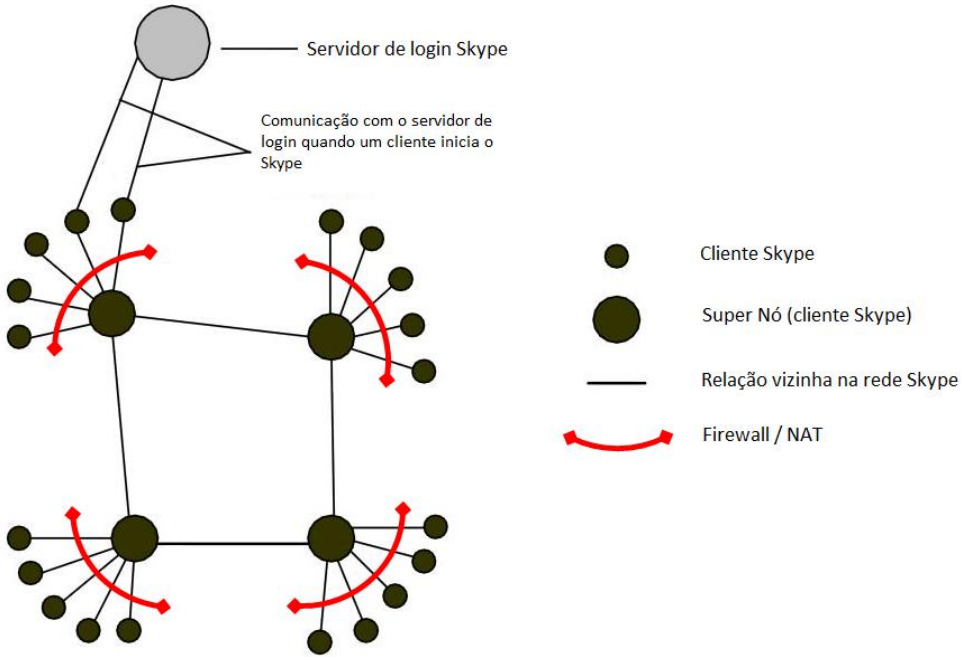


Figura 1 – Arquitectura parcial da rede Skype (Adaptada de [36]**)**

Segundo a página *web* de suporte [37] do Skype, clientes com esta aplicação activa que não estão protegidos por uma *firewall* e têm um IP público, assim como capacidade para processar ligações, são candidatos a super-nós para auxiliar na troca de dados entre clientes residentes em diferentes redes privadas, sendo impossível de outra forma comunicarem. Para evitar o barramento de tráfego por *gateways* ou outras componentes, foram implementados mecanismos neste *software* que lidam com essas questões. Devido às limitações de uma rede do tipo mencionado, um cliente Skype pode ter sérias dificuldades ou até mesmo ser incapaz de contactar todos os outros clientes na rede desta aplicação, o que levou ao desenvolvimento da tecnologia Global Index (GI). Com esta última, os super-nós interligam-se e trocam as informações dos clientes para que cada um tenha o conhecimento de todos os utilizadores do Skype.

Para manter a qualidade do serviço, o Skype mantém vários caminhos na rede para escolher o que apresenta melhores condições para o transporte de pacotes em diferentes momentos. Os dados trocados entre clientes são encriptados pelo algoritmo Advanced

Encryption Standard (AES) de 256 bits e as chaves públicas dos utilizadores passam pelo certificado RSA de 1536 ou 2048 bits, operações que são efectuadas em servidores *proxy* com capacidades para tal ou nos próprios clientes [38].

O Skype não suporta supressão de silêncio para manter as ligações que transportam media e íntegra o seu próprio *codec*, designado SILK. Este foi especialmente criado para obter os melhores resultados na Internet, e deu origem ao *codec* Opus [39]. Em conferência, o cliente com *hardware* superior será aquele que mistura o áudio e/ou vídeo ao servir de ponto central para os restantes intervenientes, deixando de sustentar qualquer conexão entre si. Por último, uma conta pode estar activa em instâncias distintas deste cliente VoIP, onde as mensagens instantâneas e chamadas são replicadas para essas instâncias.

No dia 22 de Dezembro de 2010, um conjunto de servidores dedicados à gestão de mensagens instantâneas *offline* sofreram uma sobrecarga, tendo como consequência a falha do cliente Skype 5.0.0152 para Windows e de 25 a 30% dos super-nós [40]. Para solucionar tal debilitação, a equipa do Skype introduziu múltiplos super-nós na sua rede, apelidando-os de mega-super-nós, recorrendo a serviços *cloud* como EC2 da Amazon e posteriormente aos equivalentes da Microsoft [41].

O Skype continua a ser popular e a inovar na área das comunicações em tempo real.

2.4. BigBlueButton

Este projecto teve a sua origem em 2007 por intermédio do Dr. Tony Bailetti, na altura o director do programa Technology and Innovation Management da Universidade de Carleton, e na sua vontade de fazê-lo chegar a qualquer estudante no mundo. Decidiu então desafiar estudantes a conceber um sistema de conferências que tem como objectivo oferecer aos colegas uma experiência remota e completa de aprendizagem [42]. Em Junho de 2008 o projecto teve a sua primeira versão *open-source* e lançou um canal de comunicação entre programadores envolvidos nesta aplicação. Com o passar dos anos foram observadas várias evoluções nesta ferramenta e em 2010 foi criada a fundação BigBlueButton para planear o seu futuro.

A versão actual é a 0.81 e tem os principais elementos para suportar aulas remotas, como vídeo, voz, mensagens instantâneas, diapositivos de apresentações, partilha de ecrã, *whiteboard*, gravação e reprodução de aulas [42]. O aspecto do BigBlueButton é apresentado pela Figura 2.

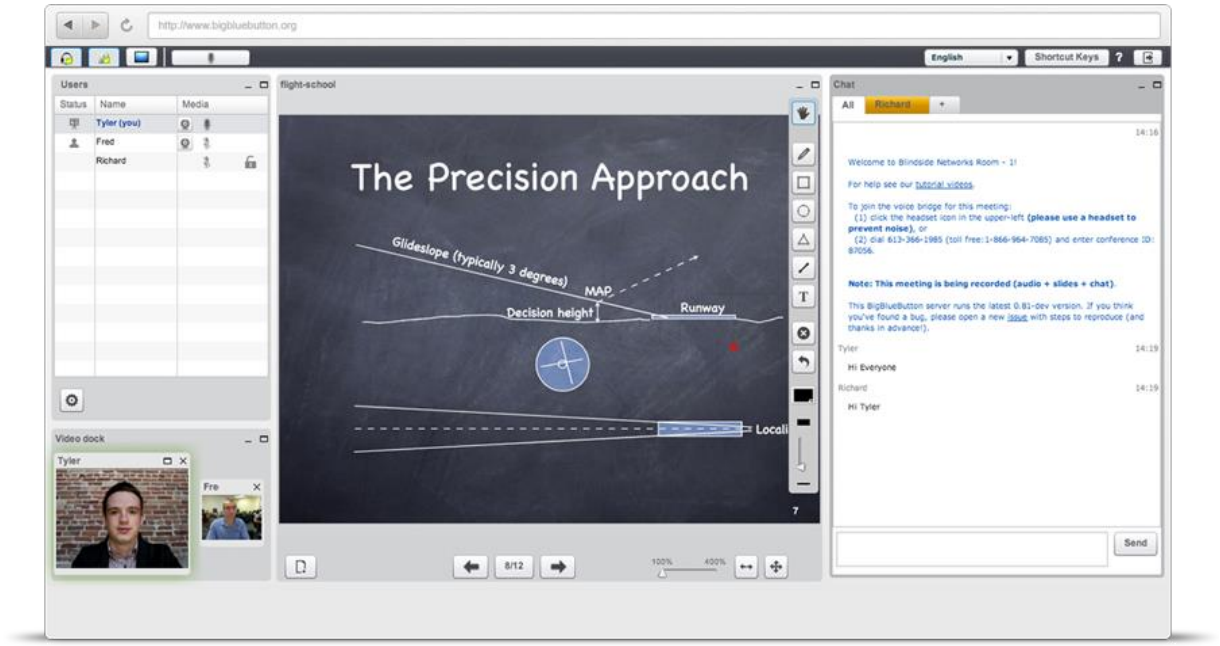


Figura 2 – Interface BigBlueButton [43]

Para tornar esta interface funcional, diversos componentes *open-source* estão interligados neste complexo sistema. A Figura 3 mostra a sua arquitectura.

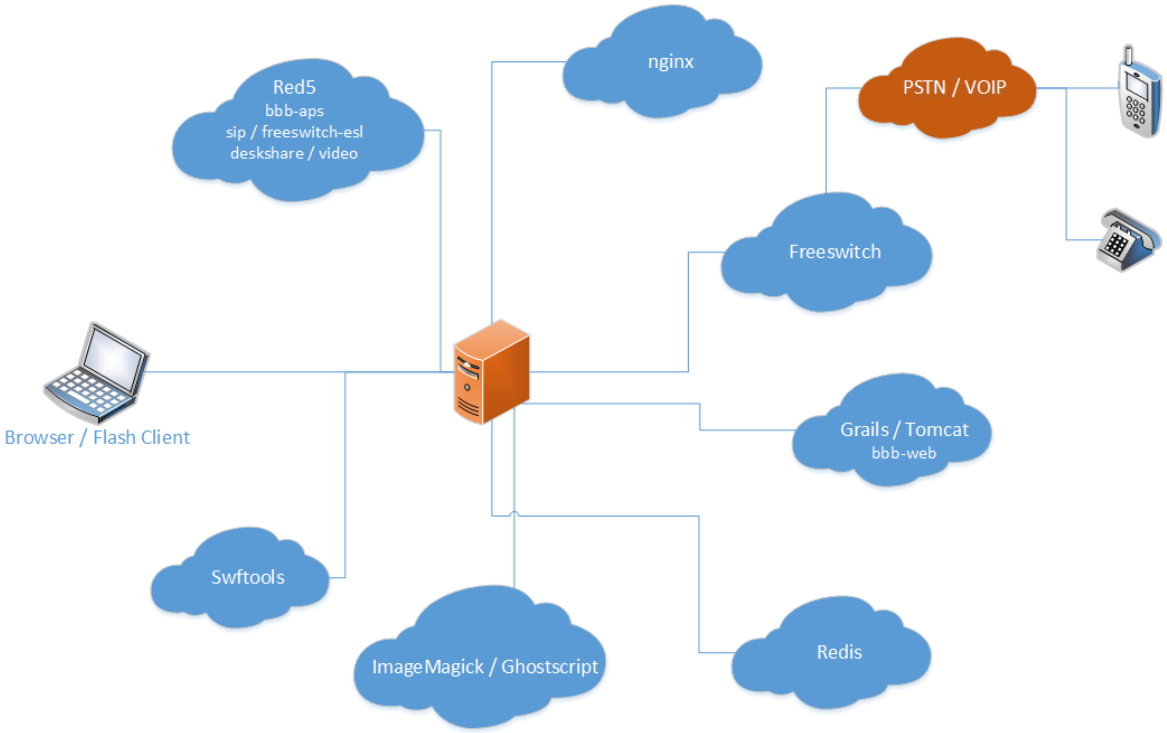


Figura 3 – Arquitectura do BigBlueButton (Adaptada de [44]**)**

A única componente que funciona na parte do cliente será a página *web* que sustenta a tecnologia Flash, sendo as restantes pertencentes ao lado do servidor [45].

Começando pelo Freeswitch, este servidor é responsável pela telefonia e gere as ligações entre participantes que se juntam numa sessão através de um dispositivo telefónico, isto é, que incorpora VoIP ou pertence a uma PSTN. Tem a habilidade para suportar diversos protocolos de áudio, vídeo e texto, assim como fazer *transcoding* de áudio sempre que necessário dos dois primeiros, mas apenas a voz é aproveitada pelo BigBlueButton [46].

O Nginx é um servidor HyperText Transfer Protocol (HTTP) com suporte aos principais sistemas operativos e diversas características, destacando-se a capacidade de integração através de módulos com outras aplicações, como autenticação efectuada por servidores de HTTP externos e manter informações temporárias de servidores com memcached [47] ou FastCGI, transmissão de ficheiros MP4 e FLV, balanceamento de carga e suporte para todos os protocolos de email [48]. No projecto discutido neste tópico,

esta aplicação tem o objectivo de fornecer o protocolo Real Time Messaging Protocol (RTMP) para os módulos bbb-web e bbb-apps, além de albergar a interface deste *software*.

Red5 é uma componente central que gere áudio e vídeo para clientes Flash. Desenvolvido em JAVA, integra o protocolo RTMP e variantes para transmissão e gravação a partir de *streams*, ao fazer uso de determinados formatos: FLV, F4V, MP4, 3GP para vídeo e MP3, F4A, M4A, AAC para som [49]. Os utilizadores do BigBlueButton são sincronizados por esta peça, garante que todos os conteúdos multimédia trocados não terão atrasos e comunica com o Freeswitch para intervenientes ligados por intermédio de um dispositivo telefónico.

O *software* Redis, desenvolvido em ANSI C, tem como foco as bases de dados em que estas são mantidas em memória ao recorrer a pares de chave e valor, com a possibilidade de guardar em disco ou criar *logs* de operações. Entre as suas características principais estão a replicação de bases de dados, sincronizações rápidas, introdução de *scripts* Lua, chaves com tempo limitado e configuração do seu comportamento para se assemelhar a uma *cache* [50]. No BigBlueButton, Redis tem a função de guardar as informações das quais os módulos bbb-apps e bbb-web se servem para comunicar entre si.

A *framework* Grails tem como objectivo a criação de aplicações *web* para o Java Virtual Machine. Groovy é a linguagem de eleição para desenvolver nesta ferramenta que promove convenção sobre configuração e Don’t Repeat Yourself (DRY) para facilitar a programação [51]. Tomcat Apache é um servidor aplicacional baseado nas tecnologias Java Servlets e Java Server Pages [52]. Em conjunto gerem a criação, agendamento e processo de autenticação das conferências do BigBlueButton.

O Swftools tem a capacidade de manipular ficheiros Flash e inclui diversos módulos para criar imagens, criar Portable Document Format (PDF), fundir ficheiros Flash ou pesquisar texto, e tem como objectivo converter as apresentações em PDF para ficheiros Flash no sistema aqui em destaque [53].

As restantes componentes ImageMagick e Ghostscript entram em acção caso o Swftools falhe nas suas funções, onde o Ghostscript tratará da manipulação dos PDFs e o ImageMagick das imagens [54], [55].

Foi elaborado um *roadmap* até à versão futura 1.0 com especificações que incluem escrita no quadro interactivo da interface, suporte ao *codec* de vídeo h.264, melhoria da escolha de dispositivos de áudio e vídeo, redução do tempo de instalação de um ambiente

para desenvolvimento, adição de uma API em JavaScript para gerir o BigBlueButton a partir de uma aplicação *web* e criação um cliente HTML5 com WebRTC [56]. Este último encontra-se em fase de criação e encontra-se disponível para quem quiser desenvolver, assim como a orientação para incorporar este elemento no BigBlueButton [57].

Foram disponibilizadas algumas imagens do cliente HTML5 final que se pretende obter, sendo uma delas apresentada na Figura 4.

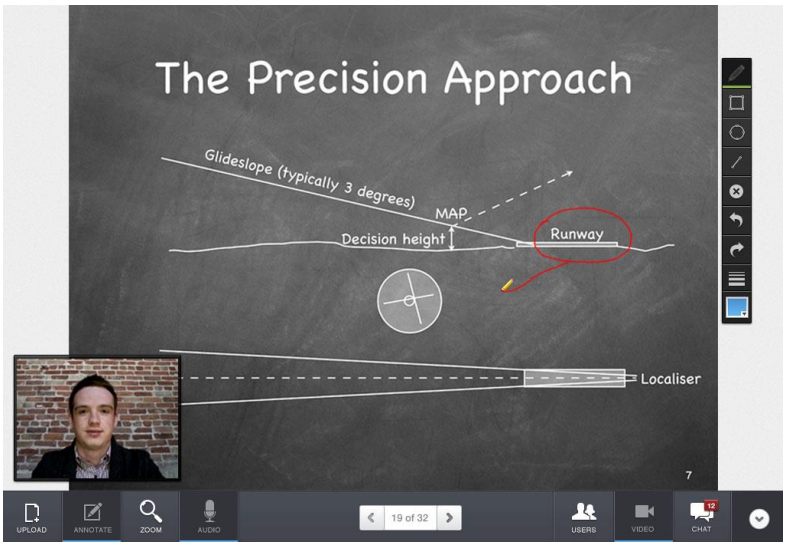


Figura 4 – Interface final do cliente HTML5 no BigBlueButton [57]

Para suportar a nova componente do lado do cliente, foi necessário alterar a estrutura da arquitectura ao adicionar novas tecnologias e essas mudanças são visíveis na Figura 5.

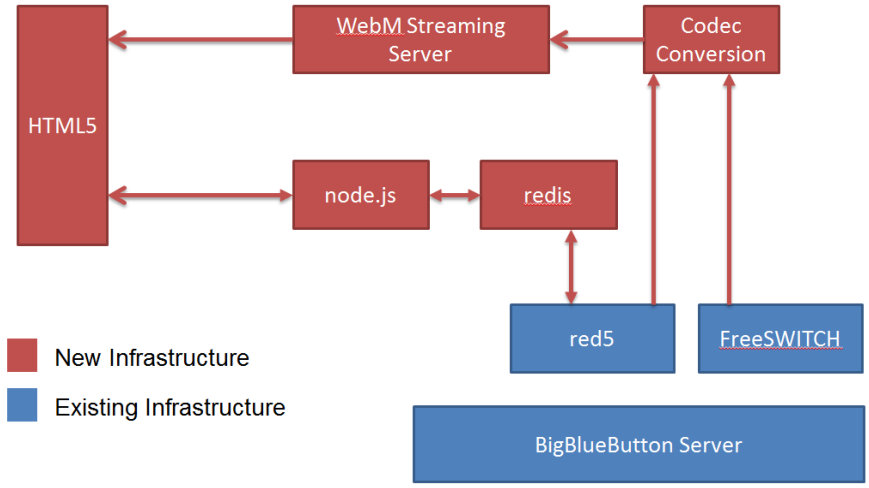


Figura 5 – Arquitectura do cliente HTML5 [57]

No cenário da Figura 5, o servidor Redis, que contém as informações partilhadas das plataformas do BigBlueButton, ganha novas capacidades para comunicar com o servidor Node.js, baseado em eventos, e o cliente HTML5 participa com este último através de WebSockets. O Red5 e o Freeswitch tratam das conversões de *codecs* áudio e vídeo para um servidor que cria os ficheiros em WebM a partir das *streams* recebidas [57], [58]. WebM é um formato aberto para ficheiros multimédia que contém vídeo comprimido pelo *codec* VP8 e o som de forma equivalente a partir do *codec* Vobis [59].

Apesar do BigBlueButton ser funcional para qualquer aula e até outros contextos, como reuniões corporativas, tem uma grande complexidade associada, dado o conjunto de componentes *open-source* que envolve para cumprir a sua missão.

2.5. Lynckia

Esta palavra é a designação de uma solução de comunicações em tempo real que incorpora a plataforma Licode, originada por uma equipa de desenvolvimento que introduziu esta aplicação. A plataforma anterior era designada Lynckia e foi construída com base na tecnologia introduzida pelo HTML5, WebRTC [60]. A equipa é constituída por três engenheiros espanhóis e lançaram este projecto com as capacidades para oferecer ligações ponto-a-ponto e conferências com gravação a partir das tecnologias HTML e JavaScript, sendo suportado no Google Chrome e Firefox [60], [61].

A sua arquitectura está organizada da seguinte forma, representada pela Figura 6.

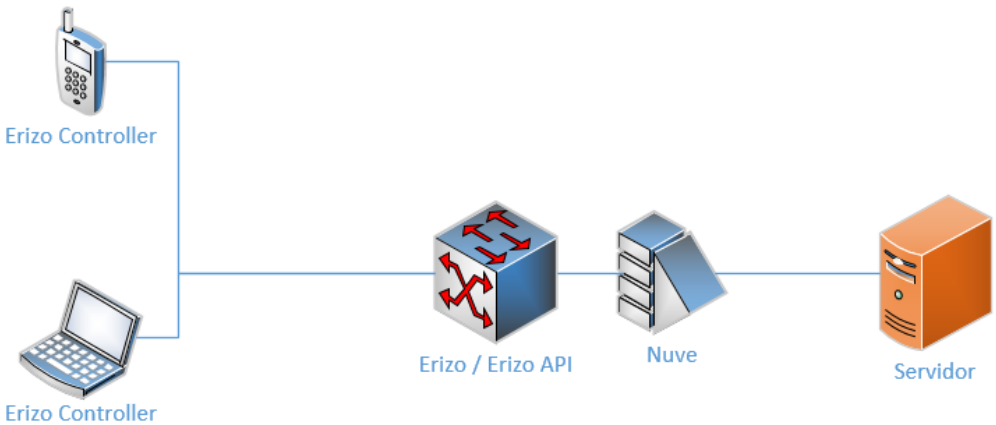


Figura 6 – Arquitectura Lynckia (Adaptada de [62]**)**

A Figura 6 mostra o cenário geral onde todas as componentes estão interligadas num sistema que concretiza vídeo e/ou áudio entre dois ou mais utilizadores, mas cada uma destas partes pode ser integrada independentemente. O servidor incorpora uma ou mais aplicações que podem integrar o serviço proveniente da Nuze API através da autenticação *token-based*, onde esses *tokens* são associados aos utilizadores da aplicação integrante, e podem ser assim geradas ou eliminadas salas Licode. A Nuve API oferece ainda escalabilidade a nível de *cloud*, controlo de acessos às salas mencionadas e registo de outras aplicações. Os utilizadores acedem a estas salas a partir de um cliente *web* que aplique a biblioteca JavaScript denominada Erizo Controller, em que esta gere os *streams* criados no *wrapper* em Node.js do MCU Erizo, a Erizo API. O Multipoint Unit Controller (MCU) tem a implementação original em C++ e é compatível com WebRTC, ao implementar os seus *codecs* e protocolos [61].

Uma versão teste está disponível ao público onde é possível fazer uma ligação com o máximo de quatro utilizadores na mesma sala [63].

2.6. WebEx

A Cisco é a detentora do conjunto de soluções aqui apresentado, sendo o mesmo capaz de gerir reuniões colaborativas por intermédio de um navegador [64]. Albergadas na *cloud* da organização mencionada, disponibilizam uma série de funcionalidades para as conferências realizadas, que incluem partilha de ficheiros, transmissão de dados multimédia, exibição de slides, *whiteboard*, mensagens instantâneas entre outras [65]. Também foi adicionado o suporte móvel através de aplicações para Android, iOS e Blackberry com as opções mencionadas [64], [66].

A WebEx oferece soluções específicas com características adequadas para cenários distintos [67]. WebEx Meetings é uma delas e tem o objectivo de criar reuniões pela *web*, onde é possível ter até cem pessoas por sessão, partilhar ficheiros, aplicações, conteúdos multimédia, ambiente de trabalho, controlo remoto de um dispositivo, *chat* e *whiteboard*. Os vídeos têm uma resolução mínima de 355x288 e máxima de 1280x720 (720p) até trinta *frames* por segundo, podendo ser expostos na interface até sete em simultâneo, sendo que um dos vídeos consegue ser expandido para ocupar completamente o ecrã. Por meio da

opção Active Speaker, um dos vídeos partilhados é destacado numa zona própria do ecrã. Com dispositivos móveis também se consegue participar nas reuniões criadas. A gravação de reuniões, integrações com Microsoft Outlook e Office para marcação de sessões, e fornecimento de áudio do cliente, além de onze línguas variadas são características presentes nesta solução. O seu suporte inclui as plataformas Windows, Linux e Mac e estão ao dispor quatro pacotes de serviços para se adequarem às necessidades de cada interessado. As informações sobre os aspectos técnicos deste sistema, como especificações das redes informáticas e o comportamento da aplicação dependendo das mesmas, são também disponibilizadas [68]–[71].

O sistema WebEx Training Center contém as características do anterior mas foca-se no contexto de aulas ou formações empresariais *online*. Para isto introduz até mil intervenientes numa aula, monitorização de atenção dos alunos participantes, opiniões através de votações, relatórios e testes, troca de mensagens instantâneas privadas ou em grupos, um esquema de perguntas/respostas organizado por tópicos e finalmente opções de pré-registos em sessões marcadas. A participação nestas sessões tem o direito de ser feita através de dispositivos Android [72].

Outro produto WebEx é o WebEx Event Center, que como o nome indica, é utilizado para organizar eventos através da Internet. A Cisco dispõe de profissionais que podem ajudar os clientes menos instruídos nestes cenários a planear as sessões para um acontecimento ou a resolver problemas técnicos. Esta aplicação especializada contém todos os elementos do WebEx Meetings e algumas funcionalidades do equivalente dedicado ao ensino *online*, sendo estas, indicadores de atenção dos participantes, *chat* privado, questões e respostas por tópicos, e *feedback* dos utilizadores. Um evento desta ferramenta consegue albergar múltiplos apresentadores e utilizadores sem conta WebEx têm a oportunidade de assistir ao próprio. Em relação à assistência torna-se exequível aumentá-la por intermédio de convites automáticos HTML e vários tipos de publicidade, com o limite de três mil participantes. Estes últimos atribuem uma classificação num parecer sobre o acontecimento, caso desejem, e o historial das participações pode ser consultado. Para além da gravação destas ocorrências, a edição e reprodução das mesmas são operações oferecidas pelo WebEx Event Center [73].

A última solução disponibilizada é WebEx Support Center, que consiste na configuração de serviços para disponibilizar suporte a clientes de uma empresa ou resolver

problemas técnicos internos de uma equivalente de forma remota, poupando tempo e custos. Uma sessão de apoio tem o limite máximo de cinco intervenientes, em que é permitida a partilha de recursos e conteúdos multimédia com outras pessoas para auxiliarem na resolução de problemáticas. Integra funcionalidades do WebEx Meetings, sistemas Customer Relationship Management (CRM) e um módulo Web-based Automated Call Distribution para encaminhar assim como gerir de filas chamadas recebidas ou efectuadas. É um sistema que ultrapassa *firewalls* de maneira segura, além de funcionar com a maioria dos navegadores *web* [74], [75]. A designação WebEx Remote Support é conjuntamente associada a este produto. Uma solução complementar à última descrita, com a denominação WebEx Remote Access, tem o propósito de aceder a diversos equipamentos informáticos sem ou com utilizadores activos, podendo desta forma realizar qualquer suporte técnico. É um sistema com grande capacidade de ligações, que ultrapassa *firewalls* de maneira segura através de encriptações SSL 128 *bits* e 256 *bits* AES. Em termos de segurança adiciona-se a gestão de palavras-chave a nível de computador, grupo e local físico, além da opção para manter o ecrã desligado aquando das intervenções aos equipamentos. Nestas últimas, estão ao dispor transferência de ficheiros, troca de mensagens instantâneas, registos e gravações de sessões, assim como redacção de relatórios posteriores. Os acessos são efectuados a partir de qualquer navegador *web* e a aplicação está optimizada para o Microsoft Windows [76], [77].

A comparação entre as características dos produtos WebEx apresentados é ilustrada na Tabela 1.

Tabela 1 – Características de produtos WebEx (Adaptada de [65]**)**

	WebEx Meetings	**Training Center**	**Event Center**	**Support Center**
Partilha de *desktop*, aplicação, documentos e *browser*	Sim	Sim	Sim	Apenas *desktop* e aplicações
Whiteboard	Sim	Sim	Sim	Não
Gravação, partilha, revisão e edição de reuniões	Sim	Sim	Sim	Sim
Ferramentas de apontamentos	Sim	Sim	Sim	Sim

Chat	Sim	Sim	Sim	Não
Transferência de ficheiros	Sim	Sim	Sim	Sim
Integração de áudio (telefone/VoIP)	Sim	Sim	Sim	Sim
Vídeo de alta definição	Sim	Sim		
Vídeo de alta qualidade	Sim	Sim	Sim	Sim
Meetings Spaces para partilhar ficheiros, calendários, documentos, notas, gravações, comentários e mensagens instantâneas	Sim	Não	Não	Não
Biblioteca de documentos com armazenamento, partilha e controlo de versões de ficheiros	Sim	Não	Não	Não
Criar e aceder a reuniões através de um dispositivo móvel	Sim	Não	Não	Não

2.7. HTML5

A tecnologia relativa ao tópico teve o seu início em 1991 e tornou-se numa parte essencial da World Wide Web, estabilizando na versão 4.01 publicada em Dezembro de 1999 [78]. Recentemente foi introduzida a recomendação para a versão 5 do HTML, definida por grupos de trabalho, um pertencente ao World Wide Web Consortium (W3C) e o Web Hypertext Application Technology Working Group (WHATWG). Apesar de não estar finalizada a aderência dos principais *browsers* foi iminente [2], [79]. Entre as versões mencionadas, muitas componentes das mesmas evoluíram e outras emergiram para atender às necessidades crescentes das aplicações *web* e a resposta para algumas delas é incluída na última versão desta linguagem, simplificando os sistemas *web* actuais. O desenvolvimento dos novos conceitos seguem directrizes que foram estabelecidas na sua concepção, onde mencionam que as novas características devem ter como a base HTML, Document Object Model (DOM), JavaScript e Cascading Style Sheets (CSS), devem reduzir dependências de *software* externo, aperfeiçoar o tratamento de erros, substituir alguns *scripts* por *tags*

HTML, ser independente de diferentes tipos de dispositivos e a evolução do desenvolvimento até ser concluído deve ser pública [2].

Das novidades apresentadas no HTML5 até à data, são realçadas as *tags* "video" e "audio" para conteúdos multimédia, o equivalente "canvas" para desenhos em duas dimensões, os objectos "sessionStorage" e "localStorage" e base de dados Structured Query Language (SQL) local para armazenamento de dados através de JavaScript, elementos para diferentes contextos como o "header", "footer", "article", "nav" e "section", novos componentes para formulários, entre eles "date", "time", "search", "calendar", "email" e "url", declaração única de páginas HTML, CSS3 totalmente suportado e remoção de elementos obsoletos, enumerando alguns, "dir", "center", "big" e "acronym" [2], [80]. Outras funcionalidades incluídas referem-se à localização geográfica, opções "drag and drop", *cache* da aplicação, possibilitam o acesso sem uma ligação à Internet activa, e eventos recebidos de recursos do servidor [81]–[84].

Verifica-se que está a ser realizado um bom esforço para evoluir a *web* de forma simples e coesa por parte desta linguagem que está a ser bem recebida, tanto pelas entidades responsáveis pelos *browsers* como pelos programadores que trabalham em HTML, e o seu futuro revela-se promissor.

2.8. Sumário

Os subtópicos apresentados ao longo deste capítulo mostram algumas das mais conhecidas aplicações que existem para comunicações em tempo real. A disposição desses pontos sugere uma evolução das funcionalidades disponibilizadas, tecnologias integradas e necessidades do mundo das comunicações. Este último ponto é demonstrado no exemplo da substituição do Windows Live Messenger pelo Skype.

Durante o seu ciclo de vida, o Windows Live Messenger apresentou diversas inovações, destacando a sua extensibilidade e integração com outros serviços, mas o seu tempo chegou ao fim. Com a chegada do Skype, determinadas integrações foram melhoradas, por exemplo com as redes sociais, e a qualidade de áudio e vídeo sofreu alterações positivas devido aos *codecs* e arquitectura deste *software*.

O BigBlueButton é um projecto que se dedica à vertente de aulas interactivas, mas contém as capacidades necessárias para se adaptar a outros contextos. Ao analisar as componentes tecnológicas que necessita para ser usufruído, a quantidade de esforço imposto para tornar este sistema numa ferramenta de eleição é tremenda.

Lynckia inclui a componente mais inovadora em relação às aplicações expostas neste capítulo, ao integrar WebRTC nos seus módulos, e disponibilizar videoconferência na *cloud*.

A Cisco detém quatro soluções distintas para se adaptar a diferentes necessidades de uma empresa. Estes produtos fornecem o maior número de opções aos seus utilizadores em relação às plataformas descritas nos subtópicos deste capítulo.

A quinta versão da linguagem HTML demonstra competências para criar páginas *web* capazes de incorporar elementos colaborativos, realçando o vídeo e o áudio, que albergam as evoluções mais significativas.

Um ponto negativo comum aos sistemas apresentados, excepto o Lynckia, é a dependência de outros *softwares* instalados no equipamento do utilizador. A excepção mencionada não fornece nenhum cliente colaborativo, concluindo-se que existe a falta de um na *web*. Ainda sobre esses sistemas, todos eles suportam VoIP, o que indica a importância deste conceito no mundo do trabalho *online.*

3. Web Real-Time Communications (WebRTC)

WebRTC é uma tecnologia que permite comunicações em tempo real, integrada nos *browsers* sem *software* adicional, como por exemplo JAVA ou Flash [85]. A sua implementação está codificada em diferentes linguagens de programação, dependendo do *software* de navegação. É acedida através de APIs JavaScript, que são definidas por propostas de *standards* do W3C [85], [86]. A equipa responsável por esta especificação é constituída por representantes de várias entidades, com especial foco no Google. Os seus esforços para tornar esta especificação num *standard web*, e o trabalho realizado com o fim de lançar a sua versão final, permitiram que fossem os primeiros a implementar por completo no seu navegador Google Chrome [87]. Este tipo de inovação contém diversas partes tecnológicas envolvidas, e em conjunto formam WebRTC.

Foram submetidos vários documentos no W3C e na Internet Engineering Task Force (IETF) com o intuito de encontrar consenso e tornar esta componente tecnológica numa parte integrante da *web* [88], [89]. Na primeira das entidades anteriores, foram apresentadas quatro propostas para definir as APIs de programação JavaScript [88]:

- WebRTC 1.0: Real-time Communication Between Browsers
- Media Capture and Streams (getUserMedia)
- MediaStream Recording
- MediaStream Capture Scenarios

No caso do IETF, os *draft*s que descrevem os protocolos empregues, a segurança envolvida e os casos de uso de WebRTC são nomeados de seguida [90]:

- RTCWeb Data Channels
- WebRTC Data Channel Protocol
- JavaScript Session Establishment Protocol
- Overview: Real Time Protocols for Brower-based Applications

- Web Real-Time Communication (WebRTC): Media Transport and Use of RTP
- Security Considerations for RTC-Web
- RTCWEB Security Architecture
- Web Real-Time Communication Use-cases and Requirements
- WebRTC Audio Codec and Processing Requirements
- Transports for RTCWEB
- STUN Usage for Consent Freshness

Para ilustrar a organização de uma implementação WebRTC, é apresentada a sua arquitectura na Figura 7.

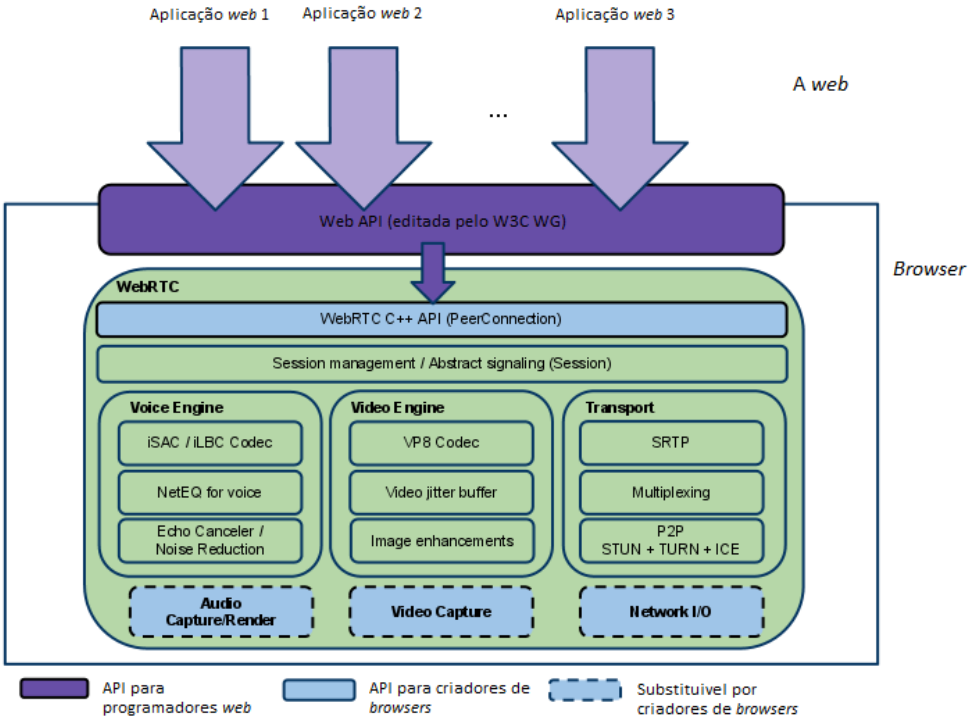


Figura 7 – Arquitectura WebRTC (Adaptada de [86]**)**

Segundo a Figura 7, cada cliente terá, pelo menos, um objecto PeerConnection associado para comunicar com outros clientes.

O processo de sinalização é o começo da comunicação e é conseguido através de um canal de comunicação bidireccional, como por exemplo, WebSockets. A decisão de qual tecnologia integrar para tal ocorrer é da responsabilidade dos programadores [86]. A informação para descobrir participantes WebRTC é trocada através do protocolo

Interactive Connectivity Establishment (ICE) e para os dados de vídeo e áudio de cada participante entra em acção o protocolo Session Description Protocol (SDP) [86], [91], [92].

As três principais componentes da implementação são o motor de voz, motor de vídeo e transporte. A primeira é responsável pelo áudio trocado entre as redes informáticas e placa de som. Contém cinco *codecs*, sendo eles: G.711, G.722, Internet Low Bit Rate Codec (iLBC), Internet Speech Audio Codec (iSAC) e Opus [86], [93]–[97]. Estes *codecs* têm uma boa integração com a *web* e em conjunto conseguem abranger um amplo espectro de frequências áudio, dando mais relevância aquelas onde se insere a voz humana, permitem a interactividade entre *streams* de áudio e tratam as falhas de voz [93]–[97]. Eles também implementam um *buffer* para compensar as perdas de pacotes das redes de informáticas e duas componentes de áudio para tratamento de voz: Acoustic Echo Canceler (AEC), para remover eco acústico do microfone, e Noise Reduction (NR), para eliminar ruídos de fundo tipicamente encontrados em sistemas VoIP [86]. O motor de vídeo processa o vídeo capturado por uma câmara através do *codec* VP8, adquirido pelo Google, com capacidades de melhoria de imagem, como a alteração da sua nitidez num vídeo. Esta parte da arquitectura também incorpora um *buffer* para manter a qualidade do vídeo ao gerir a perda de pacotes [86]. Esta estrutura tem os componentes necessários para criar sessões entre utilizadores e transmitir a informação das comunicações envolvidas [86]. Os protocolos utilizados para esse efeito são: RTP e RTP Control Protocol (RTCP) com o objectivo de enviar os dados dos clientes WebRTC, Secure Real-time Transport Protocol (SRTP) para manter a segurança dos mesmos dados, e os protocolos Session Traversal Utilities for NAT (STUN), Traversal Using Relays around NAT (TURN) e ICE para concretizar as ligações ponto-a-ponto [92], [98]–[102].

Existem duas abordagens para o tipo de arquitectura que uma solução WebRTC pode albergar. Uma das abordagens é designada de “ligações múltiplas entre clientes”, que consiste num interveniente que se conecta a uma sessão gerada por outro e por consequente uma instância do objecto PeerConnection é criada nos restantes participantes para associarem o áudio e/ou vídeo do interveniente que entrou na sessão. Neste último participante são criados tantas instâncias da mesma componente quantos os restantes participantes na chamada. Esta arquitectura pode levantar problemas de performance a nível do *hardware* pertencente aos clientes [85]. A segunda arquitectura é a “ligação única

ao servidor", e define que quando um interveniente entra numa sessão, um elemento PeerConnection é gerado e toda a informação é transmitida para o servidor. *Streams* multimédia são recebidas por um servidor dedicado a esse tipo de dados, o MCU, sendo posteriormente retransmitidas para os outros participantes incluídos na sessão, onde cada instância PeerConnection associada aos últimos mencionados recebe os conteúdos partilhados do novo participante. Por agora, estes objectos aceitam apenas uma *stream* de áudio remota, e devido a essa limitação é necessário manipular a informação referente à multimédia envolvida, para que esta arquitectura possa ser implementada. Outro problema é a dependência de servidores para tornar este cenário viável [85].

Uma das diferenças destacável entre as duas abordagens é a performance, dado que o primeiro cenário tem um peso maior nos recursos *hardware* do cliente e da sua ligação à Internet, e o segundo equivalente depende das especificações técnicas do servidor multimédia ao nível do seu *hardware*. Essa característica das implementações WebRTC é um factor que deve ser considerado devido ao consumo elevado de recursos dos dispositivos dos utilizadores, provocado pelo aumento do número de ligações numa conferência. O número máximo de ligações simultâneas sem consequências negativas para os utilizadores são dez, segundo os criadores do projecto, denominado Browser Meeting, implementado com uma arquitectura ligação única ao servidor [103], [104].

Os *browsers* de maior relevância estão a aceitar esta solução como um *standard* da *web* e prova disso é o suporte do Google Chrome, Firefox, Opera, Safari e Microsoft Internet Explorer às funções principais de WebRTC. Este último navegador requer o *plugin* Chrome Frame para incorporar as funcionalidades do Google Chrome, sendo uma delas WebRTC [86], [105].

Outra vertente que deve ser explorada é a integração com dispositivos móveis, mas a sua concretização ainda não foi completamente alcançada. Actualmente para encontrar WebRTC no mundo móvel existem as seguintes possibilidades: desenvolver aplicações nativas para cada sistema operativo móvel, servidores *gateway* que integram o protocolo SIP para sinalizar as comunicações com dispositivos móveis, ou versões móveis de *browsers*, como Bowser da Ericsson Labs e Google Chrome [106]–[108].

O *feedback* de quem explora WebRTC é essencial, como em todos projectos *open-source*, para uma evolução positiva das suas especificações e do conhecimento disponível *online*. A informação sobre aspectos técnicos pode ser encontrada no *website* onde o

código fonte está alojado, assim como alguns problemas e soluções encontradas [109]. A maioria dos problemas está relacionada com o suporte a *software* e *hardware* específicos, além de discrepâncias entre as especificações e as implementações desta tecnologia. Um exemplo do último facto está relacionado com uma situação mencionada anteriormente neste documento, em que os objectos PeerConnection deviam suportar múltiplas *streams* remotas [85]. Outra limitação relevante prende-se com a incapacidade da componente Data Channel enviar dados binários, e aqueles que podem ser enviados têm que respeitar a Maximum Transmission Unit (MTU) definida no estabelecimento da ligação ponto-a-ponto [110].

3.1. WebRTC 1.0: Real-time Communication Between Browsers

O primeiro *draft* público da especificação com a designação deste ponto foi apresentado no dia 27 de Outubro de 2011 pelo grupo *web* Real-Time Communications Working Group e ainda se encontra em discussão, sofrendo mudanças periodicamente [88]. O documento em causa indica que é atingível criar uma ligação ponto-a-ponto, com a opção de transmitir vídeo e/ou áudio entre dois utilizadores em diferentes *browsers* ou dispositivos que suportem os devidos protocolos de comunicações em tempo-real [111]. Tenta ainda definir questões para realizar o objectivo de WebRTC, sendo elas ligar dois utilizadores remotos por intermédio de técnicas transversais a NAT, tais como ICE, STUN e TURN, enviar e receber *streams* geradas num dispositivo com essas capacidades remotamente, e trocar dados variados entre utilizadores [111].

O objecto realçado na API é o RTCPeerConnection, que permite ligações entre navegadores *web* com recurso a um tipo de canal para efectuar a sinalização entre os mesmos. A sua configuração é constituída por informações de servidores STUN e TURN para acedê-los, e podem ser referenciados múltiplos dos dois géneros, em que os servidores TURN actuam também como um STUN. Este objecto contém um agente, um estado de ligação e um estado de recolha de candidatos, elementos que pertencem ao protocolo ICE. Um estado de sinalização, atributos com dados importantes para o seu funcionamento e dois conjuntos de *streams*, um relativo à multimédia local e outro com áudio e/ou vídeo

remotos armazenados em objectos MediaStream, também fazem parte do elemento principal de WebRTC [111].

Para compreender os passos de um processo de sinalização, a Figura 8 demonstra os estados de sinalização e de que forma são obtidos.

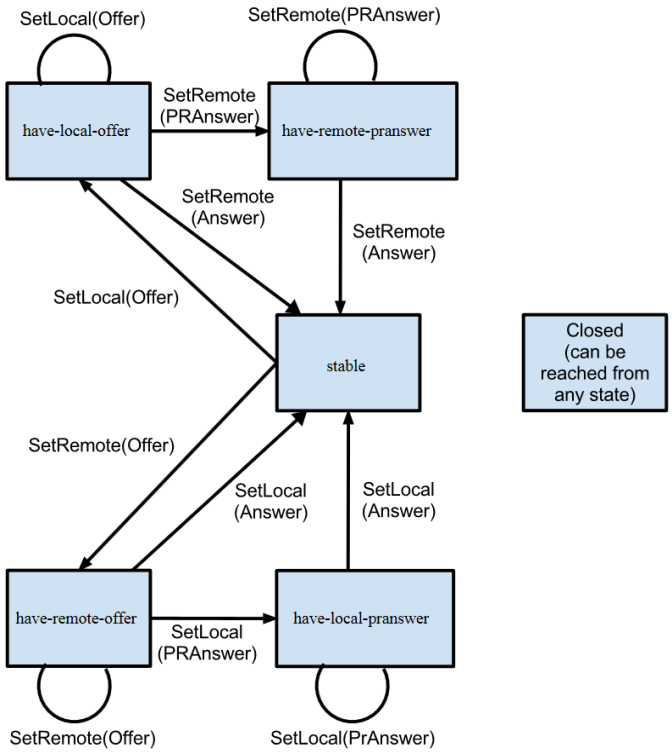


Figura 8 – Estados de sinalização e as suas transições [111]

Traduzindo estes estados para a sequência normal de eventos que iniciam uma ligação entre dois utilizadores, o resultado é o apresentado na Tabela 2.

Tabela 2 – Passos de uma chamada WebRTC (Adaptada de [111]**)**

Origem	Destino
0 - `new PeerConnection(): stable`	0 - `new PeerConnection(): stable`
1 - `setLocal(offer): have-local--offer`	2 - `setRemote(offer): have--remote-offer`
3 - `setRemote(pranswer) have--remote-pranswer`	4 - `setLocal(pranswer): have--local-pranswer`
5 - `setRemote(answer): stable`	6 - `setLocal(answer): stable`
7 - `close(): closed`	7 - `close(): closed`

Para que o cenário da Tabela 2 se componha, decorrem outros processos referentes ao agente ICE e à sua ligação para decidir o par endereço-porta por onde é trocada a informação. Os estados que uma ligação ICE pode ganhar são ilustrados pela Figura 9.

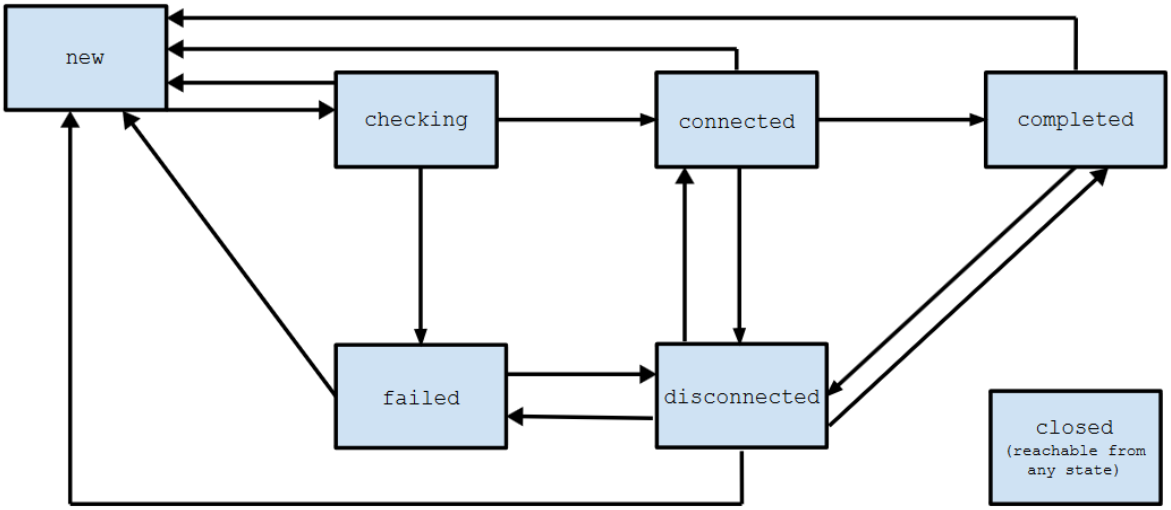


Figura 9 – Estados de ligação ICE [111]

Uma situação que envolve todas as condições da Figura 9 é exemplificada na Tabela 3.

Tabela 3 – Ocorrência a envolver todos os estados de uma ligação ICE (Adaptada de [111]**)**

Estado	Descrição
New	Em estado *new* quando é criado um objecto PeerConnection
Checking	Estado *checking*, em que os candidatos remotos são recebidos
Connected	Encontrou uma ligação válida depois do estado *checking*
Failed	A ligação falhou e foi descartada, passando novamente para o estado *checking*
Completed	Depois de *connected*, quando todas as verificações são concluídas em relação à recolha de candidatos ICE este estado ocorre
Disconnected	Após o estado *completed* e a cessão da ligação, este é o estado que a ligação toma

New	Penúltimo estado dado que pode ser transitado de qualquer outro e a recolha ICE recomeça se não for fechado de seguida
Closed	O agente ICE é terminado pela função “close”

Das questões abordadas pela especificação deste subtópico, o envio de dados arbitrários é tratado pela proposta Peer-to-peer Data API, incluída na principal. Define que uma aplicação *web* pode trocar dados genéricos através de uma ligação ponto-a-ponto ao implementar um modelo semelhante à tecnologia WebSockets. As funcionalidades desta API são disponibilizadas pelo objecto RTCDataChannel, que representa um canal de comunicação de dois sentidos. Este pode ser iniciado por intermédio de duas metodologias distintas: recurso ao seu construtor proveniente da função “createDataChannel“ do objecto RTCPeerConnection ou pelo evento RTCDataChannelEvent com um objecto RTCDataChannel associado [111]. É exequível configurar dois modos neste canal, confiável e não confiável, onde no primeiro o canal assegura que os dados foram entregues e retransmite em caso de falha. No segundo ou um número limitado de retransmissões ou o intervalo de tempo entre retransmissões permitidas pela ligação é definido [111].

Ainda no documento em questão, está definida uma API para enviar opções Dual-Tone Multi-Frequency (DTMF) entre dois utilizadores, em que é requerida a sua associação a uma componente MediaStreamTrack, objecto representativo de um canal, e esta por sua vez deve pertencer ao objecto MediaStream local. O método “createDTMFSender” do RTCPeerConnection cria esta funcionalidade [111].

Estão também incluídas no *draft* exposto neste tópico as capacidades para conhecer as estatísticas de um MediaStreamTrack, por exemplo pacotes ou *bytes* recebidos e perdidos de uma *stream* [111].

3.2. Media Capture and Streams (getUserMedia)

O documento em causa apresenta uma interface em JavaScript para aceder ao vídeo e áudio de um dispositivo com recurso a uma plataforma, tal como um *browser,* e uma API idêntica para manipular a informação gerada a partir de uma câmara de vídeo ou um microfone. A sua primeira edição foi publicada no dia 27 de Outubro de 2011 num esforço

conjunto entre os grupos Device APIs Working Group e Web Real-Time Communication Working Group, mas também em concordância com o grupo Media Capture Task Force, tendo a sua finalização prevista para o último trimestre de 2014. As suas evoluções devem seguir as seguintes regras: considerar questões de segurança em relação às capacidades e aos dados multimédia locais, realizar discussões técnicas entre grupos, basear as modificações da especificação em testes realizados e analisar opiniões por parte de utilizadores e outros grupos [112].

A Stream API mostra que o objecto MediaStream contém zero ou mais *streams* de áudio ou vídeo, sincronizadas sempre que sejam apresentadas. Estas últimas são representadas por um objecto MediaStreamTrack, que engloba a informação de um ou mais canais, dependendo do *hardware* em questão, como por exemplo um sistema de som estéreo (dois canais). A captura de multimédia é feita a partir do método "getUserMedia", ficando disponíveis os dados da entrada vídeo e/ou som para associar a elementos HTML ou guardar num ficheiro, e para estes fins é viável clonar estes dados. A Figura 10 demonstra o objectivo deste do MediaStream [112].

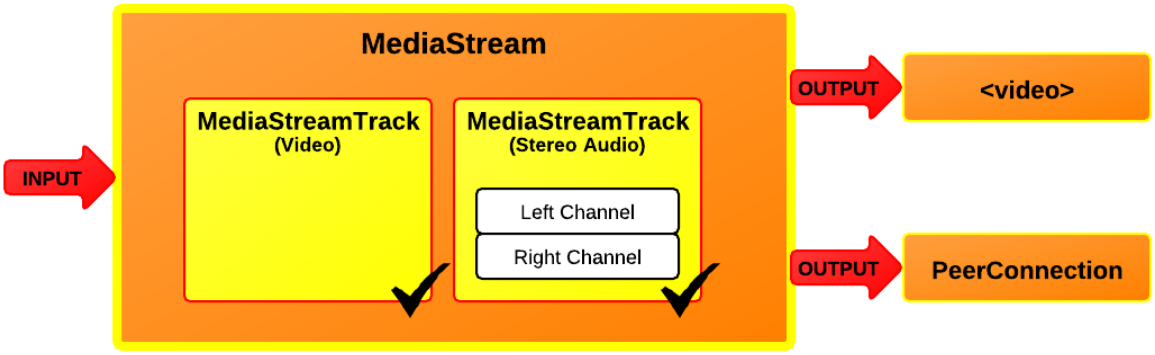


Figura 10 – Objecto MediaStream [112]

De modo idêntico, encontram-se as definições dos itens mencionados anteriormente MediaStream e MediaStreamTrack, incluindo os seus métodos, restrições e estados. Não é praticável a instanciação directa do MediaStreamTrack mas é possível criar objectos VideoStreamTrack e AudioStreamTrack, restritos a vídeo e a som respectivamente [112].

Um assunto que suscita dificuldades em ser avaliado é o modelo a seguir para ditar os comportamentos, limites e optimizações às quais as *streams* se devem sujeitar em diferentes situações. A fonte das *streams* (microfones, câmaras de vídeo) não modifica mas

as componentes que as representam podem ter requisitos distintos, seja resolução ou *frame-rate* de vídeo ou volume de áudio, não coincidindo com as restrições de cada *stream*. Para dissipar esta problemática, cada objecto MediaStreamTrack tem o método "states" que retorna um outro objecto denominado MediaSourceStates, onde se consultam as capacidades das *streams* provenientes do equipamento local. As optimizações ficam a cabo de quem implementa WebRTC [112].

Por fim são esclarecidas algumas directivas específicas a navegadores *web* para aceder ao *hardware* fornecedor de multimédia e manter o comportamento esperado das implementações [112].

3.3. MediaStream Recording

A primeira aparição pública do *draft* aludido deu-se no dia 5 de Fevereiro de 2013, na tentativa de criar uma técnica de gravação simples e com suporte para diversos cenários [88], [113]. Segundo este documento, ao instanciar um objecto MediaRecorder, o método "record" está disponível para começar a gravar até que seja chamado o método "stopRecord" ou o objecto MediaStream seja terminado. Por defeito quando uma gravação cessa é retornado um Binary Large Object (BLOB) que é codificado num tipo Multipurpose Internet Mail Extensions (MIME) definido, por omissão, na plataforma que alberga estes objectos. Há a hipótese de limitar a quantidade de dados que se recebe de uma só vez e intervalos de tempo para receber *buffers* mais reduzidos dos dados. Até à data de acesso a este documento foi verificado que os métodos mencionados, "record" e "stopRecord", não estão definidos na especificação, apenas mencionados, e em vez destes existem os métodos "start" e "stop" com os mesmos propósitos. A indicação da instância MediaStream a ser gravada, é efectuada pela passagem da mesma como argumento do construtor do objecto pertencente a esta especificação [113].

Os grupos de trabalho intervenientes da especificação "MediaStream Recording" são os mesmos da especificação apresentada no tópico anterior desta dissertação [113].

3.4. MediaStream Capture Scenarios

A primeira edição da especificação designada como o título deste subtópico foi mostrada no dia 6 de Março de 2012 com a intenção de exibir os conjuntos de cenários possíveis e impossíveis que definem as funcionalidades da API MediaStream Capture (getUserMedia), de forma a consensualizar os mesmos entre os grupos Device APIs Working Group e Web Real-Time Communications Working Group [88], [114]. O primeiro grupo deve elaborar mecanismos para descobrir a câmara de vídeo e o microfone de um equipamento, caso este tenha, e capturar elementos multimédia de qualquer dispositivo. O segundo grupo dedica-se a criar funções para processar o vídeo e/ou áudio, aplicar cancelamento de eco, sincronização de *streams*, entre outras assim como apresentar ao utilizador as *streams* por intermédio de ecrãs ou sistemas de som locais. Esta última tem contribuição da especificação HTML Media Capture, pertencente ao HTML5, e escrita pelo Device APIs Working Group [114], [115].

Os cenários encontrados no documento constituem a captura de vídeo e comunicação por mensagens instantâneas, processamento de interacções com elementos multimédia e gravação, gestão e preservação de conteúdos oriundos de variadas câmaras provenientes de um só dispositivo, gestão de conferências e de *streams* das mesmas para gravação assim como revisão posterior, e questionar a permissão para aceder à câmara e microfone de um dispositivo com o propósito de evitar situações maliciosas [114].

Ainda no *draft* descrito neste ponto, estão definidos os requisitos que os *browsers* devem implementar para suportar as permissões, *streams* locais, dados multimédia remotos e gravação [114].

3.5. RTCWeb Data Channels

O documento com a designação do tópico específica as configurações do transporte de dados genéricos, à excepção da multimédia, proveniente do protocolo Stream Control Transmission Protocol (SCTP). O transporte de dados genéricos é bidireccional e torna-se um serviço pertencente aos *browsers*, ao usufruir da compatibilidade assim como segurança do protocolo *standard* anterior por encapsulamento no equivalente Datagram

Transport Layer Security (DTLS). Este último opera sobre ICE/UDP e ganha acesso a redes que utilizam NAT [116].

Os requisitos neste *draft* envolvem o suporte para vários canais de dados simultâneos, implementação de canais confiáveis e não confiáveis (apresentado no tópico "WebRTC 1.0: Real-time Communication Between Browsers"), integração de controlo de congestionamento, segurança, execução de uma aplicação desenvolvida por terceiros e abstracção do endereço de Internet do utilizador. Os mesmos canais devem também permitir a transmissão de mensagens sem limite de tamanho além da sua fragmentação, garantir que não interfere com outros Data Channels ou *streams* multimédia paralelas e evitar a fragmentação dos dados por IP. O SCTP para além de obedecer à maior parte dos pontos prévios, no ambiente de navegadores *web*, promete albergar várias *streams* de dados unidireccionais, entregar mensagens com ou sem ordem específica e mensagens totalmente confiáveis ou parcialmente confiáveis [116].

Aplicado em WebRTC, as mensagens SDP que formam uma ligação a partir desta última tecnologia, contêm informações de controlo de congestionamento, número de *streams* SCTP necessárias e outros dados dos pontos que comunicarão entre si. Um Data Channel é definido por uma *stream* receptora de dados e outra para o envio de dados, ambas SCTP, em que o canal é gerido e iniciado pelo protocolo WebRTC Data Channel Protocol e garante que as mensagens são enviadas ou que o seu tempo de vida expira antes de terminar [116].

3.6. WebRTC Data Channel Protocol

Em WebRTC, um dos compromissos assumidos resume-se a criar uma funcionalidade para que os *browsers* comuniquem por intermédio de uma ligação ponto-a-ponto com dois sentidos, e este *draft* apresenta o protocolo para elaborar esse compromisso [117].

Numa descrição geral do protocolo, os canais são originados por associação SCTP com um conjunto de parâmetros, e são duas as formas para tal ocorrer. Uma dessas formas é o envio de uma mensagem "DATA_CHANNEL_OPEN" através de uma *stream* inutilizada da associação SCTP. Na condição de inexistência de *streams* emissoras

inutilizadas, é adicionada uma semelhante ou é retornado um erro. Este procedimento deve ser realizado no lado receptor com uma verificação da disponibilidade do canal. O outro processo para criar um objecto Data Channel requer a troca dos parâmetros requisitados por via externa, em que uma *stream* inutilizada é requisitada a este protocolo ou é indicado que uma *stream* irá ser aproveitada com parâmetros personalizados. A aplicação externa ficará similarmente responsável por garantir as últimas acções na parte receptora e a prevenção de colisões nas *streams*. As *streams* criadas desta forma podem ter opções de envio divergentes entre os dois lados [117].

Se uma mensagem chegar a uma *stream* não iniciada, esses dados recebidos serão mantidos em espera até que esta seja aberta por uma das opções previamente apresentadas neste subtópico. Caso a mensagem "DATA_CHANNEL_OPEN" seja recebida quando anteriormente existia uma pré-configuração do canal, deve ser gerado um erro. Os canais são terminados com um SCTP Reset na *stream* emissora e se acontecer o mesmo, provocado pelo utilizador numa *stream* receptora, então a correspondente emissora deverá sofrer um SCTP Reset [117].

O início da negociação é efectuado a partir da descrição de uma sessão WebRTC, definida pelo protocolo SDP, permutada entre navegadores *web*, normalmente por intermédio de um serviço de sinalização alojado num servidor. A associação SCTP é então criada com uma porção de *streams* limitadas pela aplicação *web* ou, por defeito, com dezasseis. Posteriormente é permitido o acréscimo de *streams* [117].

O envio de dados é efectuado com base em Payload Protocol Identifiers (PPID) do protocolo SCTP, diferentes dos empregues nos canais de comunicação multimédia, e é recomendado que grandes quantidades de informação a serem enviadas sejam repartidas em blocos de tamanho menor sem exceder os limites da ligação e do SCTP [117].

3.7. JavaScript Session Establishment Protocol

O protocolo definido pelo documento em questão, tem a capacidade de gerir a definição das sessões ao controlar a sinalização obrigatória para criar o suporte necessário à informação de vídeo e/ou áudio a partir de JavaScript. As tecnologias de sinalização são

abordadas posteriormente nesta dissertação e são essenciais para o funcionamento de WebRTC [118].

O JavaScript Session Establishment Protocol (JSEP) consegue, quase por completo, abstrair o navegador *web* do seu funcionamento e deve tomar cuidados quanto às limitações do mesmo, como o recarregamento de uma página que pode terminar uma ligação WebRTC activa, caso o seu estado não for preservado num servidor. As suas potencialidades são exploradas de forma simples, sendo apenas preciso uma forma de comunicar as descrições das sessões e trabalhar em conjunto com o protocolo ICE [118].

A máquina de estados e o fluxo para iniciar uma chamada foram descritos anteriormente nesta dissertação no tópico "WebRTC 1.0: Real-time Communication Between Browsers", e a máquina de estados ICE fica no *browser* porque apenas este retém os dados dos candidatos ICE e do transporte. O navegador tira partido da flexibilidade e performance que a opção de enviar separadamente os SDPs e informação de transporte oferece, característica que depende do protocolo de sinalização. Apesar da simplicidade que concede, a aplicação *web* que implementa este protocolo necessita de perceber o processo de sinalização [118].

A Figura 11 demonstra um cenário geral do JSEP.

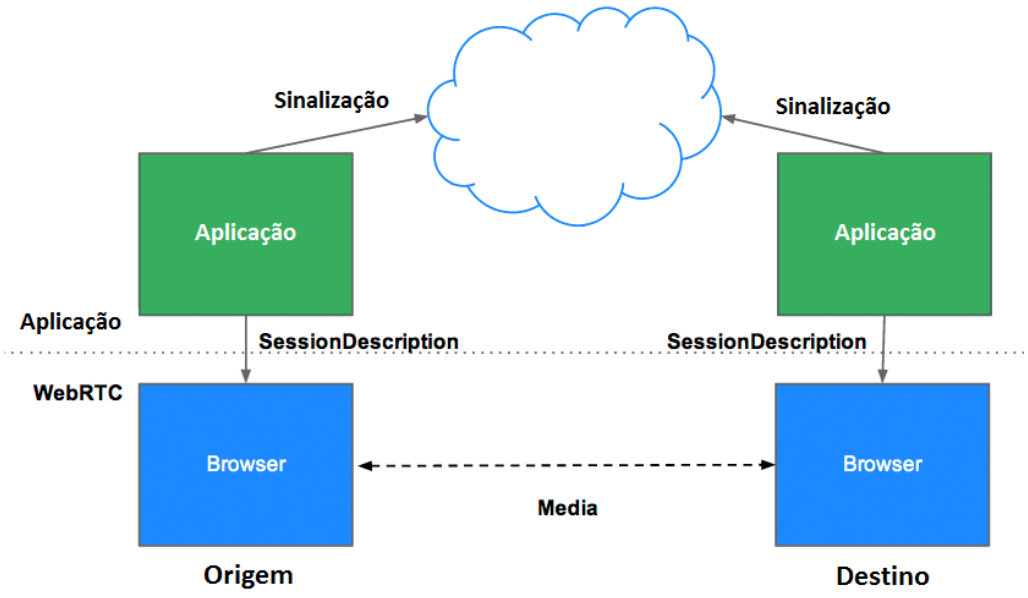


Figura 11 – JavaScript Session Establishment Protocol (Adaptada de [119]**)**

A aplicação *web* deve distinguir entre as descrições das sessões locais e remotas, as ofertas das respostas e determinar uma oferta independente de uma resposta. Para tratar os candidatos ICE, o agente respectivo notifica a aplicação, por intermédio de um *callback*, que um novo candidato foi recebido, e este é adicionado ao SDP local. Assim que todos forem reunidos, outra função *callback* é invocada para informar que esse processo foi concluído [118].

O documento apresentado neste tópico menciona o suporte da replicação de SDPs por parte dos protocolos de sinalização, em que existem dois tipos desta característica: replicação sequencial e paralela. No primeiro género, uma chamada é enviada a múltiplos pontos, onde apenas um consegue comunicar com quem ligou. A diferença em relação ao segundo tipo é que se todos os pontos em que a chamada chega atenderem, então tornam--se participantes numa conferência [118].

Na restante informação do *draft* aqui explicado, são definidos extensivamente os métodos que a API JavaScript disponibiliza e os requisitos das descrições de sessões [118].

3.8. Overview: Real Time Protocols for Brower-based Applications

O documento em questão concretiza um conjunto de funções acessíveis por via de uma API JavaScript para trocar dados multimédia entre *browsers*, onde estão inseridos os protocolos de comunicações em tempo real para esse efeito. É uma visão geral da tecnologia WebRTC e as suas componentes [120].

Nos navegadores *web* algumas características têm presença obrigatória para que a implementação WebRTC seja conseguida. Começando pelos transportes de dados, os protocolos TCP e UDP devem ser reconhecidos, e funcionalidades para estimar a largura de banda, gestão de congestionamento de tráfego e criação de ligações seguras são também requisitadas. Em relação à construção de dados, como RTP, as configurações dos mesmos, a confidencialidade e a integridade oferecidas pelos seus protocolos serão aproveitadas. A considerar identicamente são os formatos de dados e as suas especificações, *codecs*, e funcionalidades que são essenciais para se incluírem nos SDPs. A gestão de conectividade é feita pelos protocolos de sinalização, onde estes interagem com outros equivalentes e adaptam os dados no decorrer das sessões. Outro tópico descrito no documento aqui

explicado é a apresentação e controlo, em que as interacções esperadas entre participantes, como os elementos visuais do ecrã e troca imagens activada por voz, são algumas das características incluídas. Por último, as funções de suporte do sistema local como cancelamento de eco, mecanismos de autenticação e gravação da chamada, devem estar acessíveis aos utilizadores sem interferirem com as restantes funcionalidades do WebRTC [120].

3.9. Web Real-Time Communication (WebRTC): Media Transport and Use of RTP

Na especificação em causa, encontram-se os conteúdos que mostram as adaptações feitas para o protocolo RTP funcionar no plano WebRTC. A plataforma RTP contém técnicas para transmitir áudio e/ou vídeo ou até outros conteúdos multimédia, onde o seu protocolo, conjugado com várias modificações externas e estruturas distintas foi largamente adoptado nas comunicações em tempo real [121].

Contextualizando, em WebRTC são diversas as considerações a ter para que RTP se torne o protocolo transportador de multimédia [121].

Protocolos Base

RTP é um requisito em qualquer implementação WebRTC, que se divide em duas componentes, uma para transferir vídeo e/ou áudio e outra para controlar o tráfego RTP pelo protocolo RTCP. Numa sessão RTP devem estar contidos grupos de Single Synchronisation Source (SSRC) e o suporte para o envio simultâneo destas informações por um cliente. As optimizações para sessões com múltiplas SSRC são recomendadas, a escolha aleatória das mesmas para uma sessão também está definida, assim como detecção e resolução de colisões (SSRC iguais recebidas). Entre as opções avançadas destacam-se a integração para pacotes RTP e RTCP com Contributing Sources (CSRC) gerados por servidores misturadores de multimédia, envio de informação correcta para implementar a funcionalidade “lip-sync” além de suporte para as extensões de sincronização, apoiar mais do que um ponto de ligação numa sessão e intervalos de transmissão de dados RTCP, para

evitar sincronização dos mesmos. O perfil RTP aceite é o Extended Secure RTP Profile for RTCP-Based Feedback (RTP/SAVPF), baseado em extensões, onde uma delas, RTCP-based feedback (RTP/AVPF), é compatível com o perfil base RTP/AVP, presente nos dispositivos telefónicos. Os *codecs* suportados são aqueles que o protocolo RTP base implementa e estes devem negociados entre intervenientes. Uma sessão RTP terá todas as *streams* para serem enviadas de uma só vez, por UDP, e os pacotes RTP/RTCP serão simétricos, tendo facilidade acrescida ao evitar *firewalls* e traduções de NAT/NAPT [121].

Extensões

Para melhorar a performance de RTP no âmbito referido, algumas extensões são recomendadas, como:

- **Extensões de conferência:** apoio a aplicações que criem sessões multi-utilizador.
- **Full Intra Request (FIR):** é uma extensão para os sistemas que usufruem de um servidor de mistura de áudio e vídeo
- **Picture Loss Indication (PLI):** é uma extensão que ajuda a recuperar o vídeo quando este fica indisponível
- **Slice Loss Indication (SLI):** tem o mesmo intuito da extensão anterior mas genérico no tipo de dados
- **Temporal-Spatial Trade-off Request (TSTR):** é utilizada para tratar de alterações das características de vídeo pedidas
- **Temporary Maximum Media Stream Bit Rate Request (TMMBR)**

Nas extensões de cabeçalho de um RTP, as mesmas são: sincronização rápida, nível de áudio cliente para misturador, nível de áudio misturador para cliente e associação de contextos de sinalização e *streams* RTP. As extensões apresentadas são opcionais [121].

Melhorias na robustez de transporte

Neste ponto estão propostos mecanismos para manter a qualidade dos conteúdos multimédia numa chamada. Através de mensagens Negative Acknowledgements (NACK), o emissor é informado da perda de determinados pacotes RTP e modifica as características do áudio ou vídeo trocados, retransmitindo de seguida os pacotes perdidos. Um mecanismo adicional é o Forward Error Correction (FEC) que oferece protecção sobre a perda de pacotes ao manter uma largura de banda estável, podendo ser aplicado a nível de pacotes RTP ou de *codecs* utilizados [121].

Controlo de rácio e capacidade de adaptação multimédia

As tecnologias discutidas neste ponto serão transmitidas por diferentes tipos de redes informáticas e é necessário que o *software* se ajuste aos limites das infra-estruturas. Para tal é preciso implementar o algoritmo RTP Circuit Breaker e assim perceber tanto o máximo como o mínimo das taxas de *bits* pertencentes às aos dados multimédia. O protocolo RTCP também tem a capacidade de controlar o congestionamento da informação ao perceber os caminhos dinâmicos da rede por intermédio do seu algoritmo. Se um dispositivo não implementar o perfil RTP/AVPF, o sistema WebRTC deve oferecer *codecs* de baixa taxa de transmissão, para contribuir com o mínimo impacto possível na rede [121].

Monitorização de performance

Esta funcionalidade é conseguida com permuta de relatórios RTCP entre o emissor e o receptor [121].

Considerações de sinalização

O ponto em questão enumera os parâmetros mínimos que uma sessão RTP deve conter e o método de sinalização deve respeitá-los [121].

Considerações da interface WebRTC

Anteriormente foi mostrado que um objecto MediaStream é composto por zero ou mais *streams*, onde cada uma destas, caso existam, corresponde a uma fonte de dados multimédia. Estas *streams* são na sua essência informação vídeo ou áudio com uma SSRC associada, onde nesta última podem estar outras SSRCs referenciadas para retransmissão de RTP ou correcção de erros (FEC). A proposta onde este ponto é abordado informa que estas considerações podem sofrer modificações a qualquer momento [121].

Considerações da implementação RTP

Um cliente WebRTC poderá ter som e/ou vídeo de outros participantes, provenientes de um conjunto de sessões RTP, e cada uma delas tem a possibilidade de albergar uma panóplia de *streams* multimédia. Mesmo que o cenário preferível para esta tecnologia seja uma sessão RTP única, a tecnologia WebRTC deve suportar mais do que uma sessão desse tipo, com o propósito de interagir com dispositivos tecnologicamente desactualizados. Adicionalmente permitirá separar ou priorizar as *streams* multimédia com diferentes objectivos e a integração em diversos sistemas com diferentes arquitecturas, sejam múltiplas ligações entre clientes ou ligação única a um misturador de multimédia. A colisão de SSRCs iguais provenientes de dois clientes WebRTC deve ser resolvida e a sincronização de *streams* deve ser explicitada para cada uma delas [121].

3.10. Security Considerations for WebRTC

A inovação apresentada por esta nova plataforma de comunicações em tempo real envolve novos desafios em termos de segurança. A execução de um *script* no lado do cliente que faça uma chamada com intenções maliciosas é um dos casos que permite exemplificar esta situação [122].

A primeira linha de defesa para um utilizador é o seu navegador *web*, e este último é obrigado a pedir o consentimento da pessoa para conseguir aceder a qualquer componente multimédia. Isto é válido para todos os cenários identificados e para cada chamada

efectuada. Ao usufruir de DTLS, a protecção não é aplicada em cada informação gerada pelos diferentes protocolos e previne outras entidades de replicar ou construir dados comprometedores dos utilizadores [122].

Os servidores STUN têm a função acrescida de efectuar algumas verificações de segurança caso um dos terminais de ligação não implemente os protocolos ICE ou RTCP de forma a criar a ponte entre estes e WebRTC. As implementações da tecnologia mencionada podem decidir eliminar a negociação ICE inicial ou encaminhar todos os dados por um servidor TURN para o IP do cliente não ser exposto ou descoberto [122].

Para criar uma defesa contra os ataques comuns da tecnologia SIP, o mecanismo de chave pública e um serviço externo de autenticação do cliente devem marcar presença [122].

3.11. WebRTC Security Architecture

Esta proposta documenta as arquitecturas seguras para implementar uma aplicação *web* que integra WebRTC, e os requisitos de segurança pertencentes à API e implementações desta tecnologia. É assumido que as componentes da rede que o *browser* consegue identificar podem ser seguras, sejam serviços *web* de telefonia verificados por HyperText Transfer Protocol Secure (HTTPS) ou outros clientes WebRTC em que a sua origem é verificada através de DTLS-SRTP [123].

Tipicamente um sistema WebRTC contém um servidor *web* onde reside um cliente que a integra mas o número de servidores pode ser maior e podem estar em diferentes domínios. Ao não confiar na autenticação própria de uma página *web*, o mesmo sistema pode implementar um Identity Provider (IdP), seja BrowserID, Federated Google Login, Facebook Connect, OAuth, OpenID ou WebFinger. Todas as especificações em relação ao uso de um IdP são descritas minuciosamente no documento discutido neste tópico. O esquema da Figura 12 representa o cenário descrito [123].

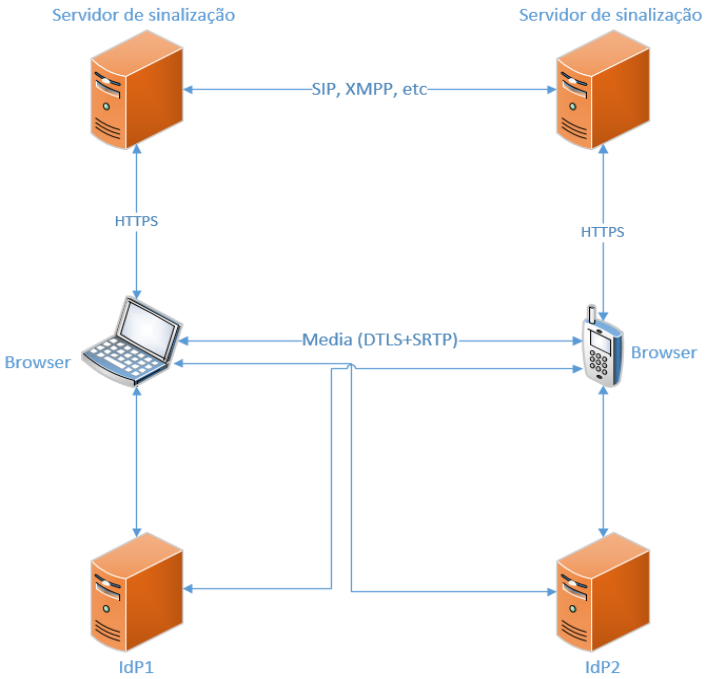


Figura 12 – Sistema WebRTC multi-domínio e autenticação com IdPs (Adaptada de [123]**)**

Uma funcionalidade JavaScript que permite realizar chamadas para clientes WebRTC, pode estar alojada numa página HTTP, e se for incluída numa outra página HTTPS, o navegador *web* do utilizador pode impedir a execução da primeira, onde irá expor um erro à pessoa que tenta servir-se desta aplicação [123].

Para além de solicitar permissão ao utilizador com a intenção de aceder a uma câmara ou microfone, também deve ser indicado quando um destes está em actividade. Se algum mecanismo esconder essa notificação então a captura terminará. O *browser* tem o dever de negar a decisão permanente para partilhar uma aplicação ou ecrã como resposta a um pedido através da API JavaScript. Este último pedido deverá ser feito em separado do pedido para aceder a componentes multimédia. Janelas que não sejam visíveis não devem partilhar conteúdos e o navegador *web* deve indicar as que estão a ser partilhadas. Outra sinalização importante é aquela que informa que o ecrã está a ser compartilhado [123].

Um requisito deduzido dos conteúdos anteriores desta dissertação é a inclusão do protocolo ICE nas concretizações de WebRTC, e nos servidores *gateway* as versões Lite e completa do mesmo têm presença exigida. O cliente deve verificar se um ponto de ligação pretendido é atingível antes de enviar quaisquer dados, por intermédio do último protocolo mencionado. A interface WebRTC deve adicionalmente oferecer a hipótese de receber

candidatos que não pertençam ao protocolo TURN para reconfigurar uma chamada activa [123].

3.12. Transports for RTCWEB

A especificação com o mesmo título deste ponto apresenta o conjunto de protocolos de transporte que qualquer implementação WebRTC deve conter, seguindo as regras de segurança estabelecidas [124].

Para transporte existem os protocolos UDP e TCP, em que o primeiro é aproveitado para ligações HTTP/Websockets, TURN/Secure Sockets Layer (SSL) e o segundo para ligações ICE-TCP. Com o objectivo ultrapassar barreiras de segurança da Internet actual, os protocolos ICE, TURN e TURN sobre TCP são necessitados, além de TURN sobre Transport Layer Security (TLS) sobre TCP poder ser incluído [124].

Os canais de dados WebRTC (Data Channels) implementam SCTP sobre DTLS sobre ICE [124].

3.13. STUN Usage for Consent Freshness

Foi verificado que antes da partilha de dados multimédia, a vontade do utilizador para tal deve ser assegurada como medida de segurança na inicialização de uma sessão, mas esta será feita também periodicamente, e o documento com o título do tópico especifica esse processo [125].

Numa ligação onde estão incluídos elementos NAT e STUN, o protocolo ICE mantém os mapeamentos NAT activos mas não é pedida uma resposta para saber se o tráfego deve continuar a ser enviado. No caso de preservar as entradas no servidor STUN essa resposta é requisitada [125].

São considerados três contadores num servidor STUN, um que define o tempo de envio de pedidos para manter as entradas STUN, outro contém o tempo que se deve esperar por uma resposta aos pedidos STUN anteriores e o último que especifica o limite temporal em que a ligação não recebe nenhum pacote. Por defeito quinze, quinze e cinco

segundos são os tempos respectivos de cada contador, e as interacções entre eles são: no intervalo de tempo imposto pelo primeiro contador é enviado um pedido para o servidor STUN da chamada e se uma resposta não for recebida em quinhentos milissegundos então o pedido original é reenviado. Este processo é repetido até atingir o tempo estimado no segundo contador e na suposição de não verificar qualquer resposta para o segundo contador ou recepção de dados para o terceiro contador, será criado um erro para que código JavaScript possa lidar com estas condições [125].

3.14. Web Real-Time Communication Use-cases and Requirements

A maior parte dos casos onde são postas em prática as potencialidades de WebRTC envolvem *software web*, mas em certos cenários são envolvidos dispositivos de outro tipo. Essas ocasiões e as questões a considerar para elas são discutidas neste *draft* [126].

Como considerações transversais a todas as circunstâncias, existem as seguintes: os clientes podem ter IPv4 ou IPv6, a rede informática de cada cliente tem um rendimento próprio, um utilizador pode estar ligado a uma rede com diferentes qualidades multimédia, as redes informáticas dos constituintes podem estar congestionadas e os utilizadores têm a hipótese de estar numa rede com NAT [126].

Os requisitos apresentados na especificação em causa foram praticamente todos relatados nesta dissertação e maior parte deles são aplicados ao primeiro caso-de-uso apresentado, que se refere a uma aplicação *web* WebRTC capaz de efectuar chamadas entre *browsers* [126]. Os restantes casos de uso envolvem páginas *web* com serviço de chamadas WebRTC entre navegadores onde um cliente está numa rede com NAT que bloqueia tráfego UDP, um *software* semelhante ao anterior onde um utilizador está ligado a uma rede com *firewall* que permite apenas ligações HTTP, uma aplicação idêntica à última num ambiente empresarial ou um serviço semelhante ao anterior em que o tipo de ligação à rede muda. Além destes acrescenta-se o primeiro caso-de-uso modificado com a introdução de QoS, uma aplicação WebRTC com a opção de partilha de ecrã, outra que implementa partilha de ficheiros, uma página *web* com WebRTC no contexto de visualização de um jogo, plataforma com suporte para sessões com mais do que dois participantes, jogos *online* com vários intervenientes e comunicação áudio, infra-estrutura

web que permite realizar ou receber chamadas para e de dispositivos telefónicos, um serviço baseado no anterior mas com chamadas para um centro de atendimento e finalmente uma aplicação que habilita videoconferência com um servidor central [126].

3.15. WebRTC Audio Codec and Processing Requirements

No documento apresentado são mostrados os *codecs* de áudio a implementar e o seu processamento em WebRTC com o intuito de promover a interoperabilidade [127]. Os *codecs* a implementar são o Opus com o formato respectivo para RTP e com a característica "ptime" até 120 milissegundos; G.711, PCMA e PCMU com um canal, taxa de 8000 Hz e "ptime" igual a 20 milissegundos; eventos de telefonia como dígitos DTMF. Se for possível processar áudio acima dos 8000 Hz, é indicado o Opus antes de PCMA ou PCMU e os clientes podem solicitar quaisquer configurações de áudio do *codec* activo [127].

O nível de áudio na parte da voz humana é considerado nas frequências acima de 300 Hz, independentemente da taxa de processamento, e deve ser ajustável para evitar distorção, com recurso a um compressor ou uma baixa no valor de ganho. No facto de gravações ou ficheiros áudio o ganho pode ser controlado pelo utilizador [127].

A implementação WebRTC tem a obrigação de englobar um mecanismo AEC ou assegurar que o som das colunas do utilizador não é captado pelo seu microfone [127].

4. Tecnologias de sinalização para WebRTC

A interoperabilidade é uma característica chave em WebRTC, e esta realidade é observada na consideração empregue nos seus documentos de especificação, permitindo tanto a liberdade dos programadores para integrarem com as tecnologias que desejam, como a expansão de contextos onde pode ser inserida. Um dos aspectos personalizáveis para quem trabalha com WebRTC é a tecnologia aproveitada para indicar o princípio de uma sessão. Este tópico apresentará algumas escolhas de forte popularidade e grande abrangência com tal intenção.

4.1. Session Initiation Protocol

SIP é um protocolo que disponibiliza a capacidade de criar, gerir e eliminar associações/sessões com determinadas características para efectuar troca de dados em tempo real entre vários equipamentos ligados à Internet. As funcionalidades descritas são adquiridas por intermédio de um servidor SIP *proxy* e a sua versatilidade é ampla na integração com múltiplos protocolos. A sua aplicabilidade reside nas comunicações em tempo real, na telefonia pela Internet (VoIP), em que as sessões transmitem dados multimédia. Nestas comunicações o protocolo SIP baseia-se na aplicação que irá ser utilizada para receber as sessões, na disponibilidade do utilizador com o objectivo de esclarecer se o mesmo está receptivo para comunicar, nas capacidades do equipamento com o intuito de determinar as competências do *hardware* pertencente ao cliente, no estabelecimento de uma sessão para negociar as características ideais da mesma e na gestão de uma sessão, de forma a tirar partido das funcionalidades de uma chamada para transferir dados, alterar parâmetros das mesmas funcionalidades e introduzir outros serviços [108].

Para tornar a telefonia pela Internet uma realidade, SIP deve ser conjugado com protocolos como RTP, Real Time Streaming Protocol (RTSP) e Megaco, para gerir *gateways* com ligação a redes PSTN, e SDP [108].

Em SIP, o fluxo de mensagens para iniciar uma sessão entre dois *softphones* é representado na Figura 13.

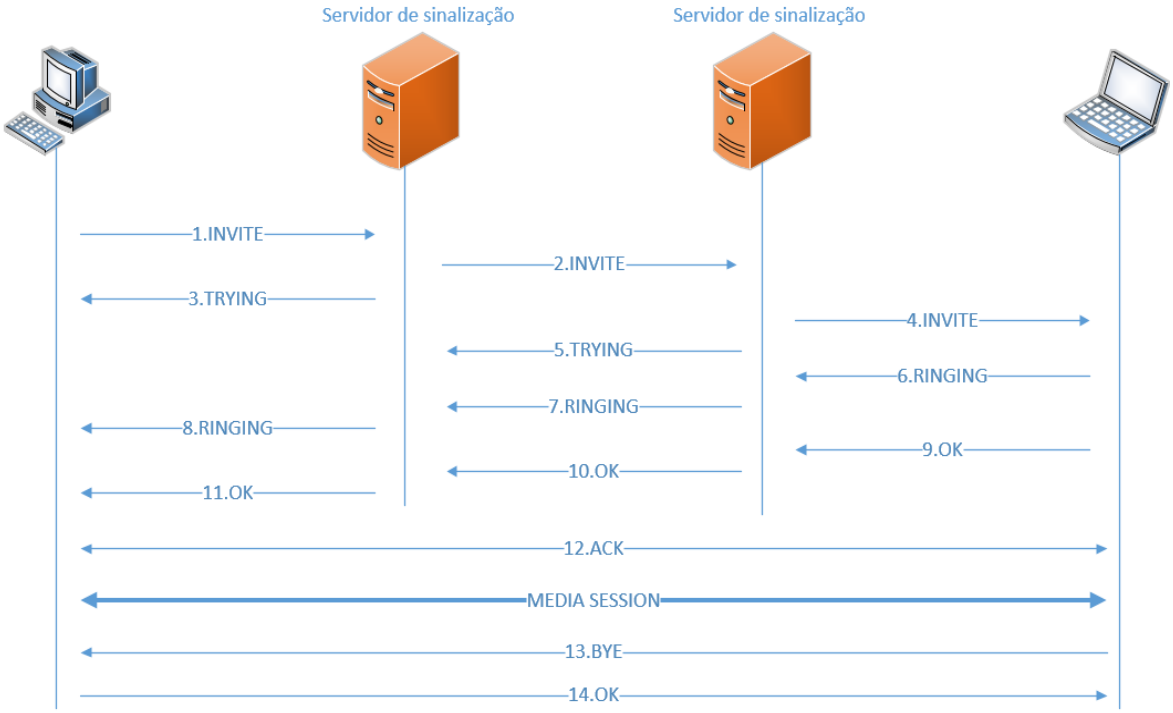


Figura 13 – Fluxo de mensagens SIP para uma chamada (Adaptada de [108]**)**

A estrutura das mensagens SIP contém dois elementos principais, o cabeçalho e o corpo, em que o primeiro é composto pelos dados referentes ao ponto a remeter e o segundo retém a informação que o emissor pretende enviar ao receptor, tipicamente mensagens criadas por protocolos diferentes. Uma mensagem com os campos mínimos no seu cabeçalho é exemplificada de seguida [108]:

```
INVITE sip:b@two.com SIP/2.0
Via: SIP/2.0/UDP pc33.one.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: B <sip:b@two.com>
From: A <sip:a@one.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.one.com
CSeq: 314159 INVITE
Contact: <sip:a@pc33.one.com>
Content-Type: application/sdp
Content-Length: 142

(corpo omitido - SDP)
```

A linha inicial apresenta o nome do método e o endereço SIP do ponto a contactar, neste exemplo é um convite para começar uma chamada. De seguida são exibidos os campos "Via", onde é mostrada a informação do endereço por onde o sistema recebe as respostas para os seus pedidos e um parâmetro "branch" que identifica a operação, "Max-Forwards", que define o limite de saltos da mensagem até ao destino, "To", onde o nome do contacto e o seu endereço SIP é encontrado, "From", que alberga os dados da origem da mensagem e o critério "tag" com objectivos de identificação, e finalmente o campo "Call-ID", onde um identificador global único para a sessão, originado a partir de um conjunto de caracteres aleatório e do IP ou nome do equipamento, é incluído. Segue-se "CSeq" (Command Sequence) constituído por um número inteiro e o nome do método, o parâmetro "Contact" que apresenta o endereço SIP para onde os receptores devem enviar as respostas, o "Content-Type" que se refere ao tipo de conteúdo do corpo, sendo uma mensagem SDP no exemplo apresentado, e por fim o "Content-Length" onde é mostrado o tamanho do corpo em *bytes*. Duas particularidades a apontar, os três campos "Call-ID", "To" e "From" combinados identificam unicamente a sessão e o "CSeq" é aumentado por cada transacção da mesma [108].

A mensagem apresentada em cima é um pedido SIP e as respostas diferem pelo campo "Status-Code" adicional, sendo este último um número inteiro com três dígitos que representa o resultado de um respectivo pedido. O primeiro dígito destes códigos indica a categoria do estado do pedido, deixando os restantes dígitos livres para criar associações entre estados específicos, relativos à categoria do primeiro algarismo. Os grupos mencionados são [108]:

- **1xx:** grupo Temporário – o pedido foi recebido e está a ser processado
- **2xx:** grupo Sucesso – a mensagem foi recebida e aceite
- **3xx:** grupo Redireccionamento – para completar o pedido é necessário executar outra actividade
- **4xx:** grupo Erro de Cliente – ou a mensagem está sintacticamente mal construída ou o servidor não pode concretizar o pedido
- **5xx:** grupo Erro de Servidor – o servidor foi incapaz de processar o pedido

- **6xx:** grupo Falha Geral – nenhum servidor tem a capacidade de resolver o pedido recebido

O protocolo SIP conta com inúmeras extensões que exploram todas as suas potencialidades, mas apesar disso ainda preserva desvantagens além de vantagens, descritas na Tabela 4.

Tabela 4 – Vantagens e Desvantagens de SIP (Adaptada de [89], [128]**)**

Vantagens	Desvantagens
Flexibilidade na integração com diversas tecnologias	Se existirem telefones SIP e SCCP na mesma rede é necessário traduzir os eventos DTMF de cada
Open-Standard logo personalizável	Implementações proprietárias de SIP
Mensagens em texto	Processamento de texto consume uma quantidade considerável de recursos
Suporte a clientes com diferentes capacidades multimédia	

4.2. Extensible Messaging and Presence Protocol

O acrónimo do título, XMPP, representa este protocolo, definindo-se como um perfil da linguagem eXtensible Markup Language (XML) que viabiliza a comunicação quase em tempo real de dados estruturados entre múltiplas redes informáticas com domínios distintos. O documento que dita este protocolo como *standard* da *web*, elabora a formação e cessão de *streams* XML, métodos de encriptação de canais, autenticação, tratamento de erros e as bases para disponibilidade de uma rede, mensagens instantâneas, pedidos e respostas [129].

A troca de informação é concretizada entre um cliente e servidor ou entre servidores, e por intermédio de pequenas mensagens transmitidas.

Os passos para criar uma ligação cliente-servidor ou servidor-servidor são iguais, excepto um [129]:

1. Encontrar o IP e porto, a partir do nome de um domínio, para efectuar uma ligação
2. Originar uma ligação TCP
3. Abrir uma *stream* XML sobre a ligação TCP
4. É recomendada a aplicação de TLS para encriptar o canal
5. Utilizar o mecanismo Simple Authentication and Security Layer (SASL) para se autenticar
6. Associar um recurso à *stream* XML (localpart@domainpart/resourcepart)
7. Enviar e receber as mensagens XML desejadas
8. Fechar a *stream* criada previamente
9. Terminar a ligação TCP

A diferença anunciada antes da apresentação destes passos, é referente ao ponto 6, onde apenas os clientes que se ligam a um servidor precisam de executar esse passo. Os endereços XMPP são denominados Jabber ID (JID), e são constituídos pela designação de uma conta, um domínio e opcionalmente um recurso, separadas pelos caracteres "@" e "/" respectivamente, como o próximo exemplo demonstra [130]:

Exemplo: `joao@dominio.com/recurso`

O protocolo XMPP é de caracter genérico mas requer estruturas específicas para a informação ser comunicada, e estas são definidas por extensões listadas na página oficial deste protocolo [131]. As extensões também podem atribuir novas funcionalidades ao mesmo, como adicionar segurança às comunicações ou estender as mecânicas de presença, com o objectivo de evitar congestionamento da rede nas tentativas de restabelecimento de ligações inesperadamente perdidas [132].

As *streams* XML são um conjunto de elementos XML inseridos na etiqueta "<stream>" com os atributos necessários para realizar a ligação desejada, e o conceito das pequenas mensagens a trocar, designadas XML Stanza, relaciona-se como o elemento

XML de nível 1 de uma *stream*. Estas são constituídas por três tipos, "<message>" para enviar uma mensagem, "<presence>" para anunciar ou renunciar a disponibilidade na rede e "<iq>" (Info/Query) que representa uma interacção pedido-resposta [129].

As *streams* emissora e receptora são reveladas na Figura 14 [129].

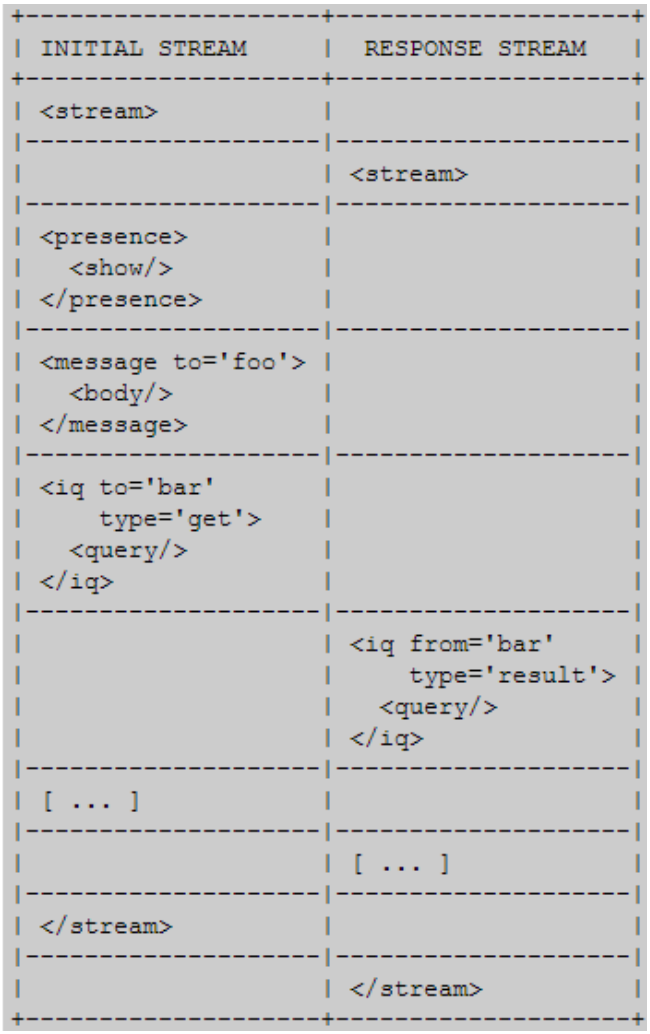


Figura 14 – *Streams* XMPP [129]

A Tabela 5 revela os atributos mínimos que o elemento "stream" deve possuir.

Tabela 5 – Valores de atributos das *streams* XMPP (Adaptada de [129]**)**

	***Stream* emissora**	***Stream* receptora**
to	JID do receptor	JID do emissor
from	JID do emissor	JID do receptor

id	ignorado	Identificador da *stream*
xml_lang	Linguagem por defeito	Linguagem por defeito
lang	XMPP 1.0+ supported	XMPP 1.0+ supported

4.3. WebSockets

O conceito do título deste subtópico define a criação de ligações bidireccionais persistentes entre clientes e servidores remotos. Esta tecnologia é constituída por uma API JavaScript, disponível em HTML5, definida pelo Web Applications Working Group, pertencente ao W3C, e um protocolo estabelecido pela IEFT [133], [134].

No documento da API, o foco principal é a definição do objecto WebSocket, em que este recebe o endereço URL onde se deve ligar e opcionalmente um conjunto dos subprotocolos pretendidos pelo cliente. Nos casos em que é feita uma tentativa de ligação a um porto seguro sem especificar no URL ou a um porto bloqueado, então um erro de segurança é gerado. Se existirem subprotocolos repetidos no conjunto definido pelo cliente ou o URL estiver malformado então um erro de sintaxe é retornado [134].

O que torna WebSockets numa tecnologia tão vantajosa para comunicações em tempo real é a sua integração em JavaScript, tirando proveito da sua filosofia de eventos e do seu processamento assíncrono. A implementação de WebSockets é simples e intuitiva ao ter dois conceitos principais: o evento "onmessage", ao qual é associado uma função para tratamento dos dados recebidos, e o método "send" para enviar a informação pretendida. Os tipos de dados texto e binário são suportados pelo objecto associado ao conceito em causa [134].

No *draft* do protocolo encontram-se as origens da tecnologia exposta neste subtópico, e na sua essência um WebSocket é uma ligação TCP com duas direcções que suporta os portos 443 e 80, assim como servidores *proxy* HTTP.

O processo de inicialização remete-se a um “aperto de mãos”, onde são transmitidas as seguintes mensagens [133]:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

A primeira mensagem constitui o pedido do cliente para uma ligação nova e a segunda é a resposta por parte do servidor a esse pedido. Analisando as duas verifica-se um conjunto de atributos fundamentais para o funcionamento esperado de um Websocket, onde a primeira linha representa o tipo de operação, HTTP Request-Line para identificar o cliente e HTTP Request-Status para o servidor, respectivamente, o “Host” retém o nome do servidor do cliente, os campos do cabeçalho “Upgrade” e “Connection” estão associados ao conceito Upgrade HTTP, “Origin” é um atributo que informa o endereço do cliente para evitar comunicação interdomínios não autorizada, os subprotocolos requisitados pelo cliente e o aquele que é escolhido pelo servidor encontram-se no campo “Sec-WebSocket-Protocol”, a chave do Websocket gerada segundo regras definidas pelo seu protocolo e codificada na base 64 está no “Sec-WebSocket-Key”, a propriedade “Sec-WebSocket-Accept” da resposta incorpora um valor gerado a partir do campo “Sec-WebSocket-Key”, pertencente ao pedido do cliente, com o objectivo de sinalizar a aceitação desse pedido, e o “Sec-WebSocket-Version” informa a versão do protocolo ao qual o cliente está a tentar aceder, onde a versão no final deste processo deve ser a 13 [133].

Os endereços WebSockets seguem a seguinte estrutura [133]:

```
"ws:" ou "wss:" "//" endereço ou nome do servidor [":" porto]
caminho para um ou vários recursos ["?"dados requisitados]
```

Exemplo: `"ws://10.0.0.0:5020/"`

A primeira notação indica que é uma ligação WebSocket e se o valor for "wss" então a ligação será segura, caso estabelecida. O porto é secundário dado que por defeito são empregue o porto 80 para ligações sem segurança e o porto 443 para ligações com segurança. O caminho deve ser representado pelo símbolo "/" se for vazio, e por opção, recursos podem ser requisitados quando seguidos do carácter "?" [133].

A acção por defeito do objecto WebSocket quando um erro se sucede na ligação é terminar a mesma. Acrescenta-se o facto de que o servidor pode findar uma conexão em qualquer altura, ao contrário do cliente que necessita de iniciar um processo para tal suceder. Com o propósito de compreender a origem do problema que despoletou esse fecho, foram criados códigos associados a causas específicas que os clientes podem enviar, e estes códigos são apresentados na Tabela 6.

Tabela 6 – Códigos de estado e razões para terminar um WebSocket (Adaptada de [133])

Código de estado	Razão
1000	Fecho normal da ligação
1001	O cliente irá sofrer uma acção (fecho do browser) que fará terminar o WebSocket
1002	Um erro do protocolo foi gerado
1003	Vinculo com o servidor cessado por falta de suporte ao tipo de dados recebido
1004	(Reservado para causas futuras)
1005	Código reservado para indicar a aplicações que esperam um código de estado, que não existe nenhum código de estado

1006	Código das aplicações para mencionar que a ligação terminou sem o processo normal ocorrer. Interdito a clientes
1007	Termina a ligação devido à recepção de tipos de dados incompatíveis
1008	O cliente fecha a ligação por a mensagem não corresponder com a política do mesmo
1009	Mensagem demasiado extensa para ser processada
1010	Indicação de negociação de extensões requeridas pelo cliente falhada
1011	Um servidor encontrou uma condição inesperada e o cliente termina a ligação
1015	Valor reservado para uma aplicação que aguarda o código estado de uma verificação de segurança falhada, por exemplo o certificado de um servidor não foi verificado

4.4. Channel API

Channel API consiste numa forma de criar ligações constantes entre uma aplicação *web* e os serviços do Google, possibilitando comunicações em tempo real para clientes JavaScript. O seu enquadramento é naturalmente encontrado em jogos multijogador, aplicações colaborativas ou VoIP [135].

Para implementar esta solução, é necessário considerar certas componentes que farão parte do sistema. A começar pela biblioteca JavaScript desta tecnologia, incorporada no cliente, esta componente tem a responsabilidade de se ligar a um Canal quando receber um elemento específico Token do servidor, de escutar eventos do Canal e enviar mensagens para o servidor, em que o último propaga para os clientes pretendidos. A componente central, o Servidor, tem o papel de criar e gerir os canais solicitados por parte dos clientes, gerar e enviar os Tokens aos últimos mencionados, e processar, assim como reencaminhar, as mensagens recebidas. O Identificador do cliente é um objecto que pode ser um *cookie*, um número aleatório, um conjunto de credenciais de login ou qualquer elemento que tenha a capacidade de ser único, para o servidor conseguir identificar as suas ligações. A entidade mencionada anteriormente Token autoriza a ligação do cliente com o servidor e tem um tempo de vida de duas horas. Um Canal neste sistema compõe uma ligação

unidireccional ao servidor, onde o último envia as mensagens de outros clientes por intermédio deste objecto. O elemento Mensagem é utilizado pelos clientes para comunicarem com o servidor através de pedidos HTTP, para posteriormente ser reencaminhado até ao seu destino. O limite de tamanho de cada mensagem é 32K. O objecto integrante Socket alberga os eventos de um objecto Canal, e o conceito Presença define a habilidade do servidor em requisitar uma notificação quando um cliente liga ou termina um Canal [135].

No *site* oficial desta conceptualização encontram-se vários exemplos da sua integração numa aplicação *web* [135].

4.5. Web Messaging

A ideia por detrás de Web Messaging baseia-se na comunicação envolvendo duas páginas HTML em domínios *web* distintos. Para tal são apresentadas na sua especificação duas formas de tornar esta ideia uma realidade, integrada na versão 5 da linguagem HTML [136].

Uma das hipóteses reside na comunicação com uma página de certo domínio, contida numa outra de domínio desigual. A cada janela estão associados os dois elementos cruciais para implementar esta capacidade: evento “message” e função “postMessage”. Um modelo em JavaScript que demonstra tal cenário é assim apresentado [136]:

```
var o = document.getElementsByTagName('iframe')[0];
o.contentWindow.postMessage('Hello world',
'http://b.example.org/');
```

Hipoteticamente, um documento HTML A tem um elemento “<iframe>” carregado com a página *web* “http://b.example.org/” e o código apresentado em cima, envia a mensagem “Hello world” para a página do “<iframe>” [136].

Num *script* do documento B, tendo este documento o endereço do "<iframe>" do documento A, está associado o evento "message" para tratar as mensagens enviadas por diferentes *sites* [136]:

```
window.addEventListener('message', receiver, false);
function receiver(e) {
  if (e.origin == 'http://example.com') {
    if (e.data == 'Hello world') {
      e.source.postMessage('Hello', e.origin);
    } else {
      alert(e.data);
    }}}
```

A segunda forma para comunicar entre páginas de diferentes domínios envolve o envio e recepção de mensagens através de um canal próprio. Especificamente trata-se de um objecto MessageChannel, que é constituído por dois elementos MessagePort, onde cada um dos últimos representa um cliente. Para entender os seus detalhes, um exemplo é exposto de seguida [136]:

```
var channel = new MessageChannel();
otherWindow.postMessage('hello', 'http://example.com',
[channel.port2]);
 channel.port1.postMessage('hello');
channel.port1.onmessage = handleMessage;
function handleMessage(event) {
  // message is in event.data
}
```

O primeiro passo é criar uma instância do objecto MessageChannel e de seguida enviar um dos objectos MessagePort (port2), através do primeiro mecanismo descrito neste tópico. Assim que a página destino recebe o objecto referido, a partir do método "postMessage" do MessagePort local (port1) são enviadas mensagens, e para receber as mesmas mensagens pelo MessagePort remoto (port2), é associada uma função ao seu evento "message". Uma nota importante em relação as dados trocados é o facto de eles poderem ser estruturados [136].

4.6. Resumo

O capítulo 4 descreve dois protocolos de sinalização de sessões e três tecnologias para as comunicações entre sistemas. O protocolo SIP demonstra capacidades alinhadas com a Internet, dado que a sua concepção foi baseada na mesma. Este protocolo foi largamente adoptado no mundo das comunicações em tempo real, com excelentes resultados. O seu concorrente aqui exposto, XMPP, mostra uma versatilidade imensa, consequência das suas variadas extensões, mas não promete comunicações imediatas.

No grupo das tecnologias para comunicar, WebSockets contém as características necessárias para se adaptar ao contexto das comunicações em tempo real, mas a sua implementação deve ser cuidada, pois a ligação ao servidor é constante. O Channel API consiste num sistema completo de comunicação, onde estão incluídos servidores do Google e objectos próprios que devem estar implementados num projecto *web*. É uma solução que pode ser inserida em contextos aplicacionais distintos e a sua arquitectura está definida, mas podem levantar restrições na sua aplicabilidade. Outra questão a considerar é a privacidade dos dados transmitidos através desta metodologia. Sobre a tecnologia final apresentada no tópico aqui resumido, Web Messaging, percebemos que dois *websites* têm a capacidade de comunicar sem nenhuma ligação activa a um servidor. Apesar da inovação, encontra-se facilmente as suas limitações, tanto ao nível de funcionalidades como ao nível de aplicações onde pode ser integrada.

5. Aplicação colaborativa final – WebRTC Experiment

As soluções aplicacionais que se encontram no capítulo referente ao estado da arte desta dissertação, apesar de populares, contêm pontos negativos particulares e universais, realçando a necessidade de instalação de *software* externo no equipamento local. Em termos tecnológicos, assistiu-se a uma evolução interessante constituída por novas ferramentas baseadas em paradigmas inovadores, com potencialidades inexploradas. A conjunção de HTML5, WebRTC e uma das tecnologias de sinalização promete revolucionar as comunicações em tempo real. Acrescenta-se o cuidado demonstrado pelos grupos de trabalhos envolvidos na concepção das duas primeiras tecnologias anteriores para introduzir compatibilidade com diversos contextos, protocolos, *frameworks* entre outros, e abranger as subáreas de comunicações em tempo real respondendo às suas carências. Chamadas à distância de um clique através de qualquer dispositivo com suporte aos *standards* da *web* tornam-se viáveis e liberaliza as metodologias para desenvolver aplicações com este fim.

O intuito de WebRTC é disponibilizar meios para transmitir multimédia entre *browsers* mas as condições para estender a outros mundos, nomeadamente sistemas VoIP e telefonia tradicional, estão reunidas na própria inovação. O projecto BigBlueButton tem planos futuros para este panorama mas a sua arquitectura envolve várias componentes e torna-se complexo de instalar. O elemento crucial para tornar este caso-de-uso possível é um servidor VoIP de forma a interligar WebRTC, telefonia pela Internet e PSTNs.

5.1. Requisitos

No âmbito desta dissertação será desenvolvida uma aplicação *web* onde três ou mais participantes dispõem de uma série de funcionalidades com fins cooperativos. Fazendo da teoria apresentada neste trabalho, a base de conhecimentos para a componente prática deste elemento de avaliação, as decisões de requisitos e implementação serão baseadas nessa mesma base.

O ponto de partida para o levantamento de requisitos é a definição das funcionalidades inseridas na aplicação. A partir da última mencionada, um utilizador deve conseguir criar sessões envolvendo áudio e/ou vídeo com outras pessoas. Estas podem estar registadas no sistema onde o cliente se insere ou podem ter algum número telefónico associado a um dispositivo, criando a lógica de conferências no caso de envolver três ou mais participantes. Um interveniente de uma conferência pode receber e fazer chamadas para outros utilizadores, menos para os que estão numa conferência WebRTC Experiment. Sempre que um novo utilizador da aplicação *web* a desenvolver entra numa sessão criada previamente através dessa mesma aplicação, todos os participantes que a usam para comunicar conseguirão trocar dados multimédia com o novo participante. Caso um interveniente desligue a chamada de outro semelhante, então este último utilizador sai da conferência. Os participantes também poderão trocar mensagens instantâneas, desenhar num *whiteboard* disponibilizado ou partilhar ficheiros entre si. Os efeitos replicativos de uma conferência do sistema a desenvolver aplicam-se similarmente aos ficheiros compartilhados e aos esboços do painel de desenho. As possíveis operações no cliente a criar são ilustradas pelo diagrama de casos-de-uso da Figura 15.

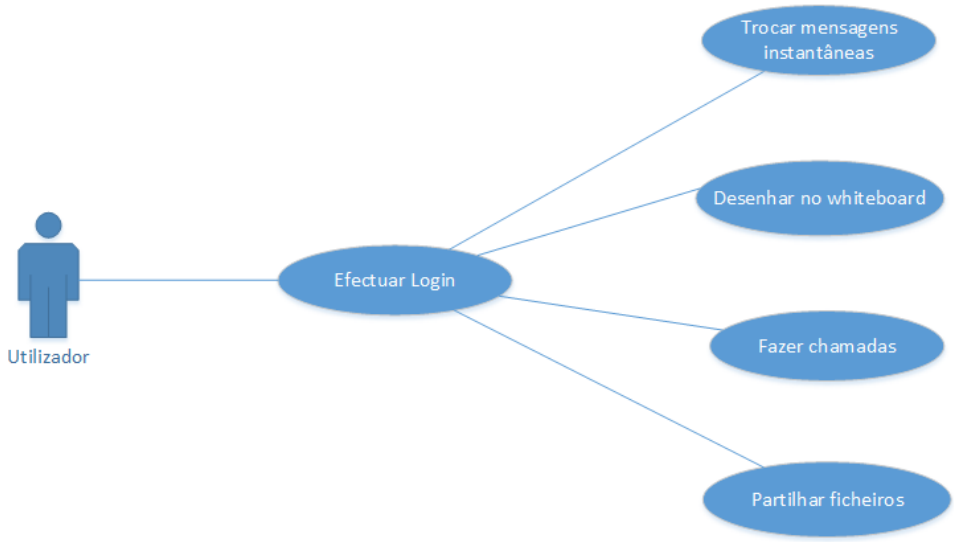


Figura 15 – Diagrama de casos-de-uso

Aquando do login do utilizador com um nome não utilizado, o interveniente pode executar as acções demonstradas na Figura 15, excepto partilhar ficheiros, algo disponível

apenas quando o utilizador tem pelo menos uma chamada activa com outro idêntico registado no WebRTC Experiment.

A interface será adaptável aos ecrãs com a resolução mínima aconselhável de 1280x720. O aspecto visual de uma aplicação *web* pode ditar o seu sucesso perante o público. Sendo esta uma plataforma que poderá ser usufruída por praticamente qualquer pessoa, as características visuais em foco serão a simplicidade e o minimalismo. Estas são duas das bases para *websites* cumpridores das regras de usabilidade. Um utilizador necessitará apenas de um dos principais *browsers*, nomeadamente, Microsoft Internet Explorer, Safari, Google Chrome, Firefox ou Opera, assim como uma ligação à Internet estabelecida, portanto a implementação do cliente proposto será apoiada em *standards web*. A estrutura será um ponto relevante nesta aplicação, ao separar a componente visual da componente lógica de funcionalidades e dos comportamentos associados.

Consentindo as premissas relatadas neste subtópico, foram então elaborados os modelos para orientação do *design* da interface, a Figura 16 mostra o ecrã para realizar o login.

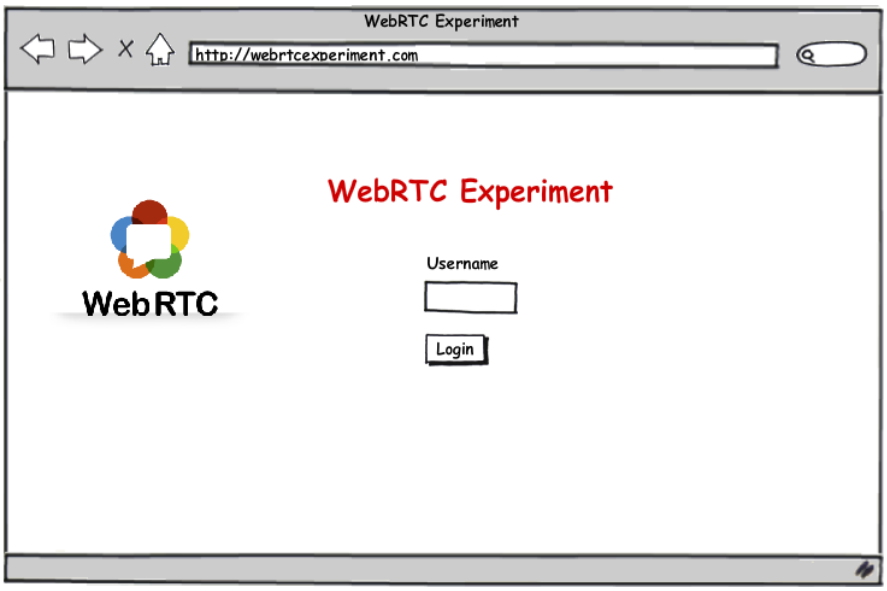


Figura 16 – Modelo de login da aplicação WebRTC Experiment

No sucesso do login, o utilizador encontrará um visual semelhante ao da Figura 17, onde poderá executar as operações definidas pelos casos-de-uso.

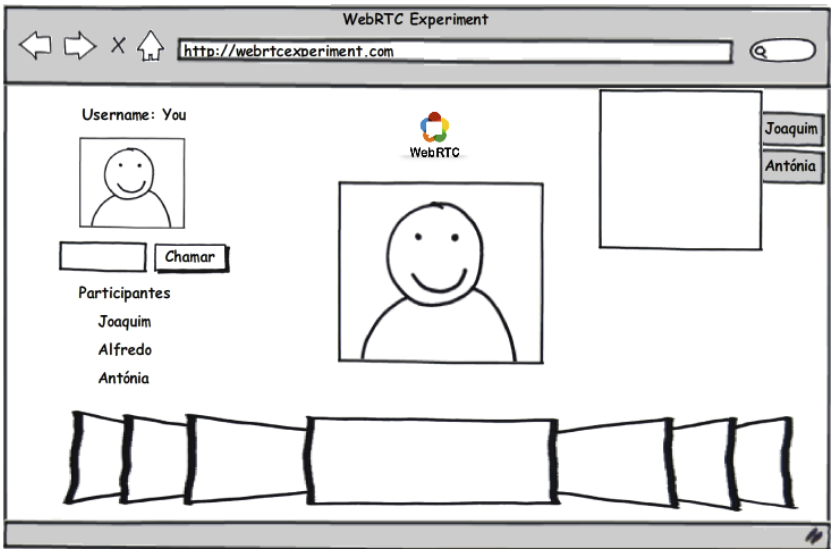


Figura 17 – Cliente WebRTC Experiment

5.2. Arquitectura

Segundo os requisitos identificados no subtópico anterior e o estudo revelado pelos capítulos desta dissertação, a parte central deste sistema será um servidor que constituí a ponte entre diferentes ambientes. Para que WebRTC consiga interagir com um servidor de telefonia, o protocolo SIP é o melhor candidato para a comunicação entre os dois mundos. Com o intuito de implementar uma lógica adicional sobre interacções do cliente e do servidor SIP, e das conferências WebRTC Experiment será criado um *gateway*.

O esquema da arquitectura final é demonstrado pela imagem da Figura 18.

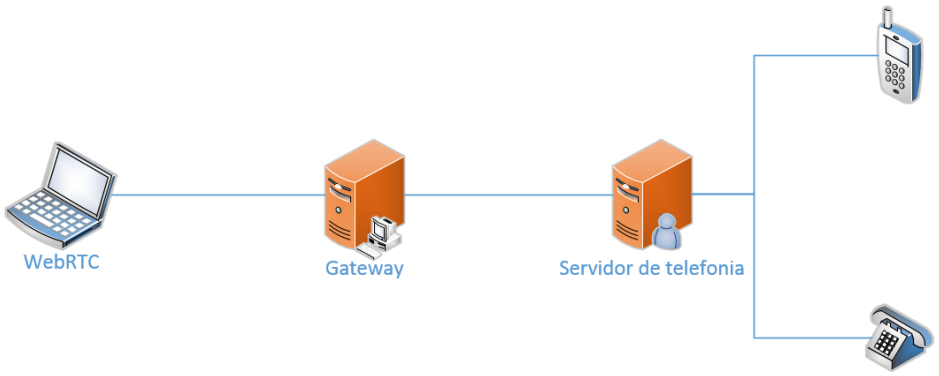


Figura 18 – Arquitectura do sistema WebRTC

Após esta fase de definição da arquitectura, as decisões de implementação, relacionadas com as ferramentas a incorporar, foram tomadas e o seu desenvolvimento foi iniciado. Com base no estudo formado, é perceptível que HTML5 e WebRTC devem ser as fundações do cliente *web*, mas com diversos desafios provenientes das mudanças ainda prováveis destas tecnologias. Numa pesquisa efectuada sobre a interligação entre as últimas apresentadas e SIP, é facilmente encontrada a biblioteca JavaScript, JsSIP [137]. Simples de manipular, com implementação completa de SIP e suporte a WebRTC, esta foi escolhida para o cliente ser capaz de se relacionar com o servidor de telefonia. Uma característica apreciada neste elemento é a implementação de SIP sobre WebSockets, da qual será o tipo de ligação para as comunicações requeridas. O aspecto negativo de JsSIP é a compatibilidade apenas com o Google Chrome, o que inviabiliza um dos requisitos propostos anteriormente [138].

As tecnologias que estabelecerão a interface e as suas funcionalidades num *browser* foram seleccionadas, faltando as opções para os servidores. Essas opções são conhecidas do projecto BigBlueButton, sendo Node.js aproveitado para o servidor intermédio (*gateway*) e Freeswitch para o servidor VoIP. Justificando tais escolhas, deve-se ponderar os papéis que cada parte assumirá baseando-se na sua performance, analisada posteriormente no subtópico 5.4.

Na implementação do sistema proposto, as componentes estão organizadas segundo a representação da Figura 19.

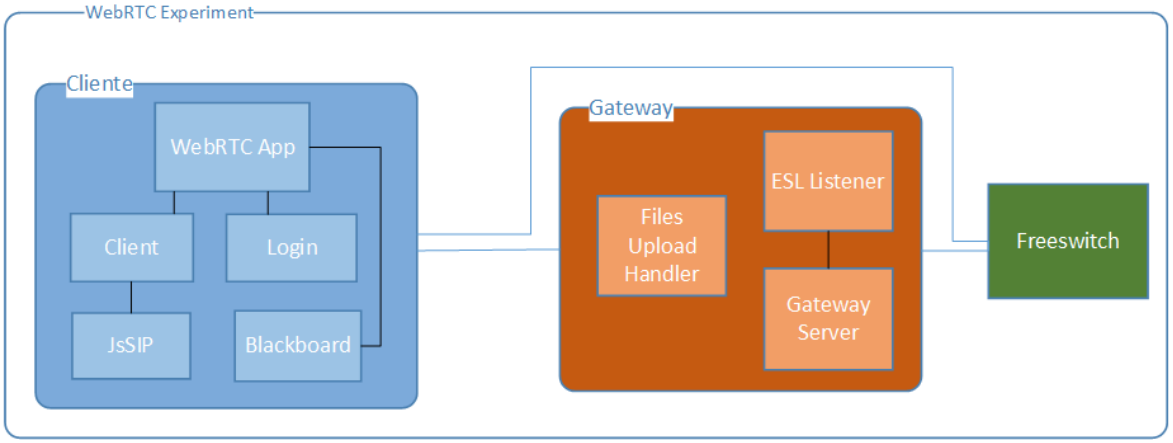


Figura 19 – Componentes da aplicação proposta

O Cliente abrange cinco bibliotecas JavaScript, JsSIP, WebRTC App, Login, Client e Blackboard. A WebRTC App lida com as comunicações entre o *gateway* e o cliente para concretizar a lógica de conferência além dos avisos gerados para o utilizador. É o "ponto central" do cliente e comunica com praticamente todas as restantes bibliotecas. O Login encarrega-se do processo de login, validando com o Gateway o nome introduzido pelo utilizador. Ao efectivar o login entram em cena os três ficheiros restantes para atribuir as funções definidas nos requisitos a uma só página *web*. O *script* Client tem a responsabilidade de interagir com o servidor intermédio para gerir algumas informações da conferência e instanciar objectos provenientes da biblioteca JsSIP, de forma a processar chamadas directamente com o Freeswitch. A JsSIP alberga a formação das mensagens SIP para enviar ao servidor de telefonia, lida com os dados das sessões originadas e integra WebRTC. Por último, o Blackboard trata das funções do painel de desenho, ao enviar os dados, resultantes do desenho que um utilizador cria, para o Gateway, que os distribui aos demais participantes. Suporta ainda o apagar completo do mesmo painel, assim como o ajustamento do tamanho do *whiteboard* em relação ao ecrã.

A entidade Gateway é composta por um *script* PHP e dois semelhantes em Javascript, em que os últimos incorporam uma biblioteca denominada "node-esl" [139]. Esta última disponibiliza a capacidade de receber eventos gerados no Freeswitch e enviar comandos para controlar o mesmo, algo exequível devido à biblioteca nativa deste servidor

de telefonia, Event Socket Library [140]. A componente ESL Listener escuta os eventos resultantes dos pedidos de sessões, cancelamento de chamadas e registo de contas SIP. Para garantir o sucesso total na captura de tais eventos por parte da última componente, a lógica de processamento das informações e gestão de conferências é executada no Gateway Server. Quando esta componente é iniciada, as contas SIP internas disponíveis são requisitadas ao Freeswitch, por intermédio do módulo "node-esl" mencionado, e armazenadas em memória no Gateway Server. Assim que é recebido um pedido de login, se o nome do utilizador for aceite, então o último elemento mencionado do Gateway fornece os dados das contas SIP ao Cliente para este se registar no servidor VoIP. Os registos e os cancelamentos enviados ao Freeswitch são monitorizados pelo Gateway Server, de forma a evitar problemas neste ponto. A componente PHP tem como função receber os ficheiros a partilhar, através de pedidos XMLHTTPRequest, e serão disponibilizados nas conferências, armazenando-os no servidor.

O Freeswitch fica a cargo de lidar com as chamadas encarregando-se da criação, conservação e destruição destas, bem como da negociação dos seus conteúdos multimédia e concretização do *transcoding* de áudio, caso seja essencial para a comunicação entre os participantes. Como foi referido, o Freeswitch beneficia ainda das capacidades que o permitem ser a ponte entre redes VoIP e PSTNs. Integra também as especificações ideais para a aplicação proposta neste documento, tomando de exemplo a implementação de SIP sobre WebSockets, o suporte a WebRTC, os diversos *codecs* de áudio e vídeo, a integração de protocolos para transmissões em tempo real e a instalação nos sistemas operativos mais conhecidos. As contas SIP, compostas por um número e uma palavra-passe, devem ser configuradas previamente para que a plataforma WebRTC Experiment seja capaz de funcionar.

O servidor VoIP Freeswitch reside numa arquitectura modular, na qual os módulos são desenvolvidos em linguagem Lua. As configurações são armazenadas em ficheiros XML, como *dialplans* e perfis SIP. Os primeiros mencionados constituem as regras para as chamadas efectuadas que devem passar pelo Freeswitch e os segundos são as configurações SIP para números internos e externos ao sistema. A funcionalidade SIP sobre WebSockets fica disponível através da associação do porto 5066 à mesma num perfil

SIP. O *dialplan* e os perfis SIP são as componentes de suporte para o processamento de chamadas, como demonstra a Figura 20.

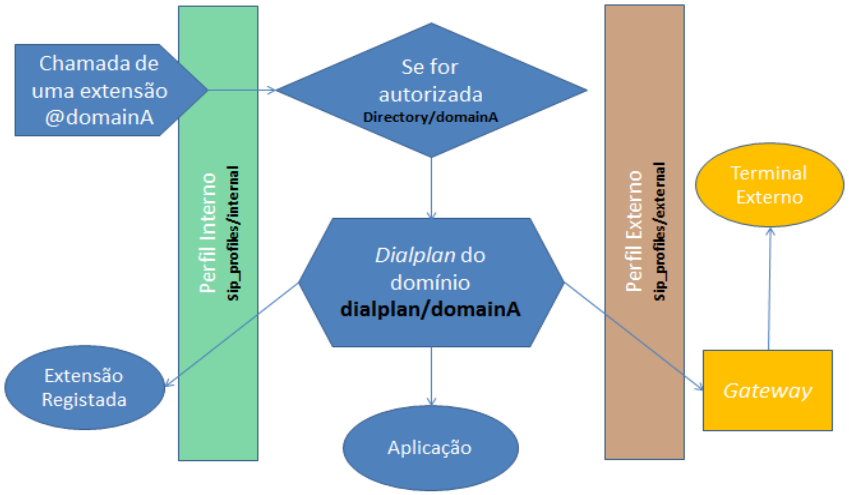


Figura 20 – Processamento de uma chamada Freeswitch

A Figura 20 mostra uma chamada iniciada por uma conta SIP interna registada no domínio A, e logo que o servidor autorize essa chamada, é computorizada pelo *dialplan* respectivo ao domínio para decidir se é entregue numa extensão relativa a outra conta SIP interna, numa aplicação ou num número exterior. A última opção é disponibilizada por intermédio de um *gateway* pertencente a uma operadora telefónica.

Sobre o Node.js, é sabido que se baseia no motor V8 do Google Chrome, e o seu segredo para o sucesso entre os outros servidores reside no seu modelo de processamento, apresentado na Figura 21 [58].

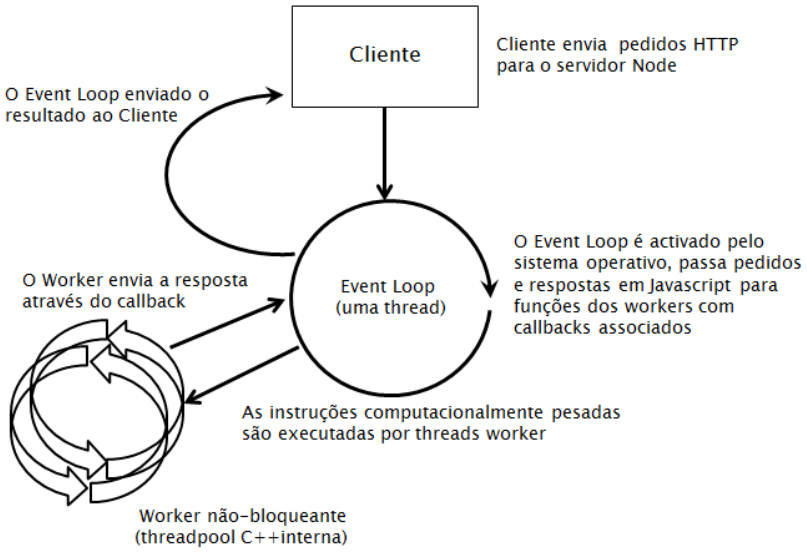


Figura 21 – Modelo de processamento Node.js (Adaptada de [141]**)**

Dado que o Freeswitch é responsável pelas sessões, a lógica de login e conferência é lidada pelo Node.js. A Figura 22 mostra as interacções entre as três partes do sistema quando uma chamada é criada no cliente WebRTC Experiment.

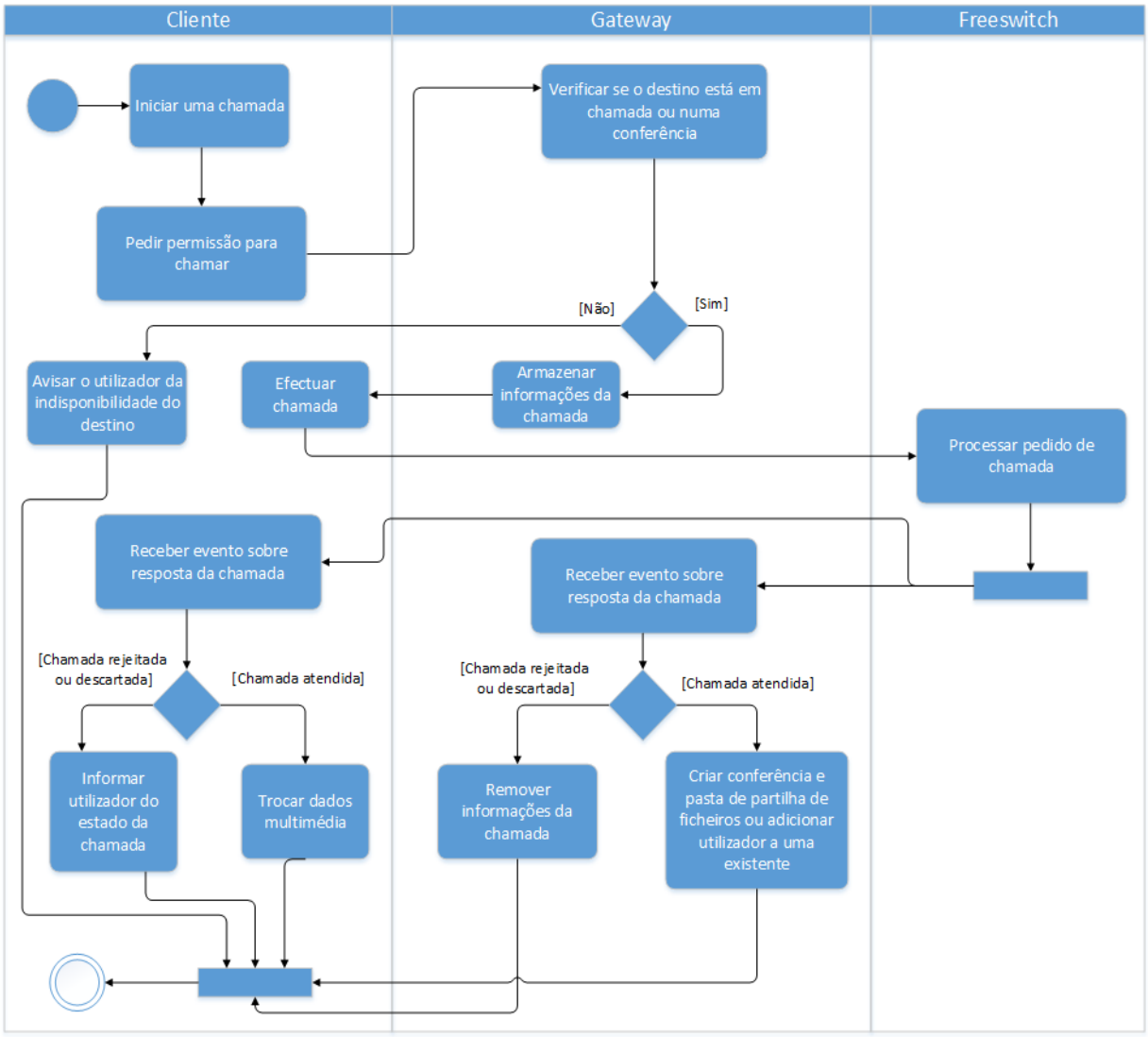


Figura 22 – Diagrama de actividade de uma chamada WebRTC Experiment

As sessões iniciadas noutro ponto SIP com destino a um cliente da aplicação proposta não são consideradas pelo servidor Gateway.

Os passos necessitados pela aplicação desenvolvida para disponibilizar os ficheiros, que um utilizador pretende partilhar com quem está na sua conferência, encontram-se espelhados na Figura 23.

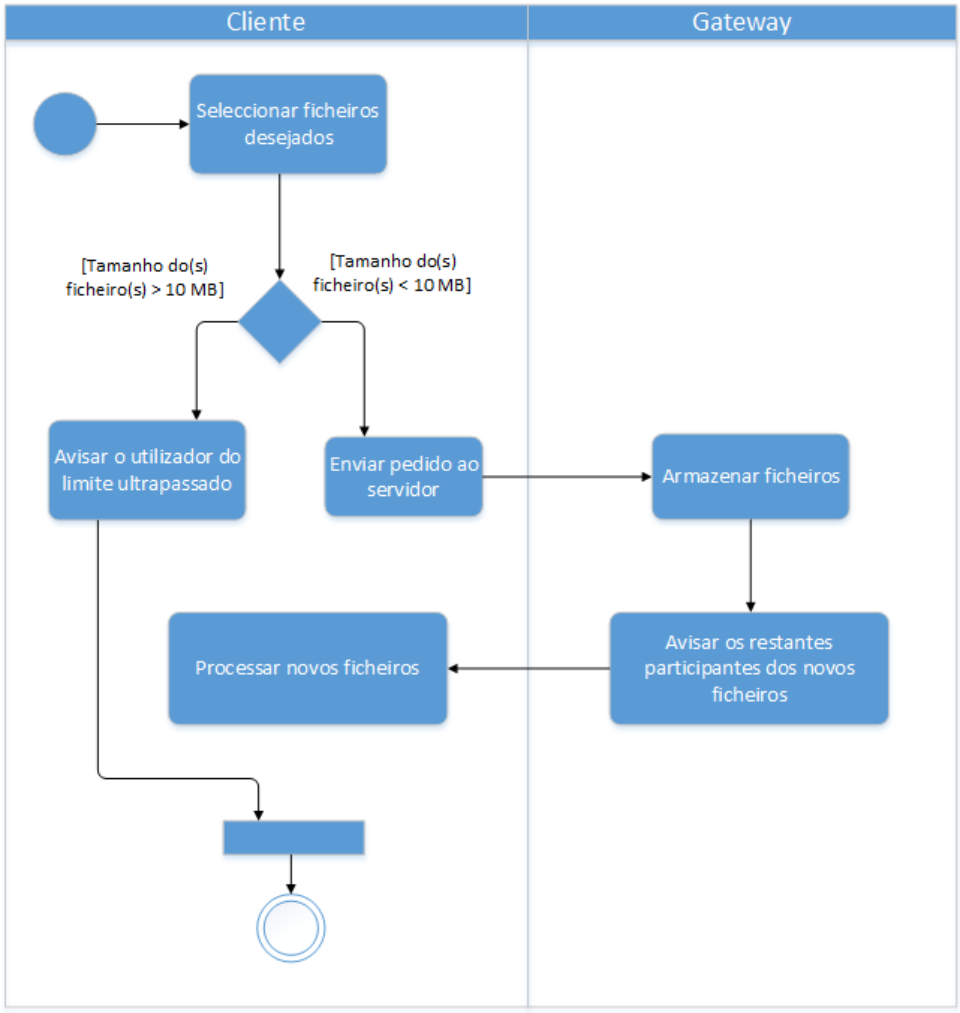


Figura 23 – Diagrama de actividades de uma partilha de ficheiros

Com estas considerações sobre as tecnologias centrais e experiências analisadoras de performance demonstradas no ponto 5.4 desta dissertação, a escolha das mesmas revela-se promissora e são merecedoras do risco assumido para criar um sistema assente nelas.

5.3. Protótipo

Ao seguir os pontos definidos nos requisitos sobre a interface gráfica do cliente, envolveram-se os *standards web*, apesar do funcionamento da aplicação estar restrito ao Google Chrome, devido à biblioteca JsSIP. Com o intuito de tornar a interface visualmente apelativa, mantendo-a simples, foram adoptadas a *framework* Bootstrap e uma extensão sua designada Colorpicker [142], [143].

Como se verifica na Figura 24, os elementos gráficos obrigatórios na página de login são uma caixa de texto e um botão para validar o nome introduzido nessa caixa quando o utilizador desejar entrar no cliente.

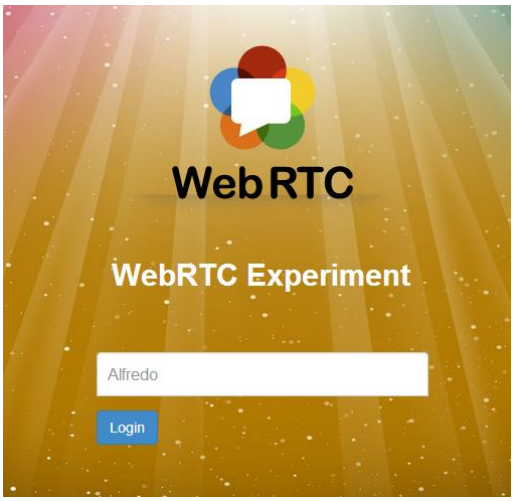


Figura 24 – Interface login WebRTC Experiment

Se o nome estiver em uso ou o campo para preencher com este se encontrar vazio, assim que o botão para validar é pressionado, um erro é exibido ao utilizador. O nome escolhido pelo utilizador deve ser constituído apenas com caracteres alfanuméricos, hífens e *underscores*, com o limite mínimo de três caracteres. A imagem de fundo apresentada é da autoria de [144]. As imagens com o símbolo de WebRTC, foram aproveitadas de [145] e [146]. A imagem associada aos elementos HTML “<video>” que apenas albergam áudio, é proveniente de [147].

A página do cliente contou com algumas transformações, resultantes dos testes de usabilidade, apresentados posteriormente neste documento. Na versão beta, logo que a

inserção de um nome fosse validada, o painel do utilizador, ilustrado pela Figura 25, era exibido.

Figura 25 – Painel do utilizador do cliente beta

Para entrar em contacto com outra pessoa, bastava introduzir o nome na caixa de texto apresentada e pressionar o botão "Chamar". Na situação de tentar efectuar uma chamada para alguém que não está registado ou no caso de o espaço para introduzir o nome do contacto estar vazio, uma mensagem de erro era exposta ao utilizador. Quando a sessão é criada com sucesso, uma janela de conversação surgia no canto superior direito, com o objectivo de trocar mensagens instantâneas com a pessoa contactada.

A versão final da solução produzida, apoiada nas sugestões registadas nos testes de usabilidade, é demonstrada na Figura 26.

Figura 26 – Interface final do protótipo

A maior alteração entre versões do cliente provém da disponibilização de um *whiteboard* e uma zona de partilha de ficheiros. Para originar uma chamada basta indicar qual o número ou o nome de utilizador a contactar. Se o mesmo rejeitar ou não estiver registado no servidor VoIP, então uma mensagem é apresentada ao utilizador. Sempre que a aplicação necessita de mostrar determinada informação pertinente ao utilizador, as mensagens geradas surgem abaixo do painel do mesmo, como a Figura 27 ilustra.

Figura 27 – Aviso utilizador não está registado numa tentativa de chamada

Por norma, quando uma pessoa se encontra numa aplicação que incorpora WebRTC, antes de trocar dados multimédia numa sessão, o navegador *web* em uso pede permissão à mesma pessoa para aceder à câmara e ao microfone do seu dispositivo. Este pedido de permissão é facilmente ignorado ou rejeitado, o que impede o funcionamento da aplicação. Para ultrapassar esta situação, WebRTC Experiment é acedido por intermédio de uma ligação HTTPS, o que faz com que o requisito de aprovação para aceder aos elementos multimédia do equipamento seja apresentado apenas uma vez. A resposta a esse requisito é guardada pelo *browser*.

Na Figura 28 verificam-se todas as funcionalidades da plataforma produzida destacadas.

Figura 28 – Interface WebRTC Experiment

Em relação à Figura 28, na zona dos participantes encontram-se dois intervenientes incluídos na conferência, em que um apenas permuta o áudio e o outro envolve vídeo e áudio. Estas ligações são estabelecidas com sucesso por influência adicional do servidor STUN público "stunserver.org", incluído nos parâmetros do objecto criado pela biblioteca JsSIP [148]. O painel do *chat* é constituído por tantas janelas quantos os participantes da conferência, com o objectivo de trocar mensagens instantâneas entre eles, por intermédio do método "Message", pertencente a uma extensão do protocolo SIP [149]. Na última imagem referenciada, a janela de conversação minimizada está destacada pela cor verde, indicação de que o respectivo participante enviou uma mensagem instantânea.

Quando um dos utilizadores de uma conferência usufrui do *whiteboard*, as suas acções neste último são replicadas no espaço equivalente para esboços dos outros intervenientes.

Relativamente à partilha de ficheiros, múltiplos objectos podem ser escolhidos e enviados numa só operação para o servidor, desde que o tamanho total do (s) ficheiro (s) não exceda o limite de dez *megabytes*. Qualquer interveniente de uma conferência pode guardar os ficheiros no seu dispositivo e eliminá-los da conferência.

5.4. Testes

No subtópico em destaque são apresentados os testes que o sistema sofreu, os seus resultados e as consequentes conclusões. Este ponto refere três tipos de experiências, às quais as componentes do WebRTC Experiment foram submetidas: testes de carga, testes de atraso e testes de usabilidade.

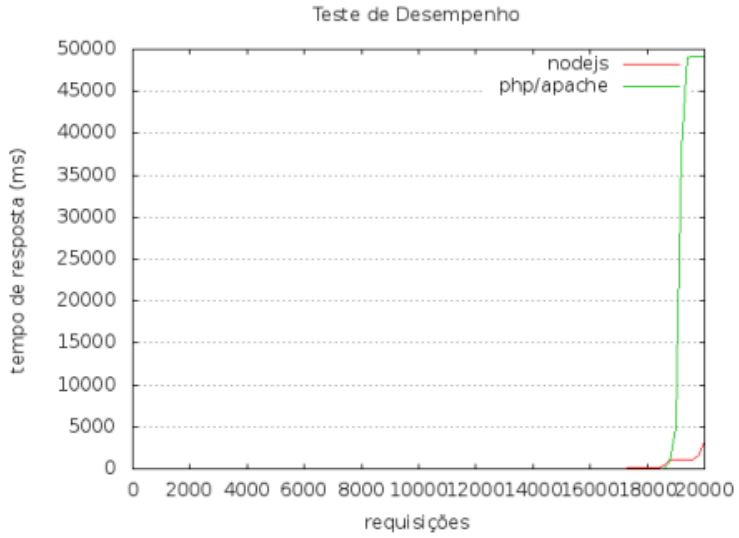
Testes de carga

Ao lidar com interacções em tempo-real, a rapidez de execução é a questão central numa infra-estrutura. O Node.js é uma resposta à altura dos desafios desse contexto, como a revista Acta Brazilian Science [150] relata nos testes comparativos entre PHP num servidor Apache e o seu semelhante baseado em JavaScript. Os excertos de código da Figura 29 respondem a pedidos HTTP de igual forma em ambos os servidores, e os resultados dos testes são demonstrados na Figura 30.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
```

```
<?php
        echo "Hello World!";
?>
```

Figura 29 – Servidores de teste Node.js e PHP/Apache [150]



DADOS	NODE.JS	PHP+APACHE
Nível de Concorrência	1000	1000
Tempo Total	4.061 segundos	49.219 segundos
Requisições Completas	20000	20000
Requisições Falhas	0	2427
Requisições por segundo	4924.91 segundos	406.34 segundos
Tempo por requisição	203.049 milissegundos	2460.973 milissegundos

Figura 30 – Resultados de testes de performance PHP/Apache VS Node.js [150]

A diferença é notória, não deixando dúvidas que Node.js é uma *framework* pronta para as comunicações em tempo real, e revela inclusive as capacidades da linguagem JavaScript no motor V8.

Foi realizado um teste extenso ao Freeswitch que totalizou cem horas de execução, em que foi imposto um limite de mil chamadas simultâneas e outro de cinquenta chamadas originadas num segundo, em que em cada uma das chamadas, um ficheiro de áudio de trinta segundos era reproduzido [151]. O resultado mostrou que mais de dez milhões de sessões foram criadas, onde 99,9% foram completadas com sucesso, deixando perto de mil e quinhentas chamadas falhadas e oitenta canais que nunca cessaram [151]. O servidor de telefonia aludido é a base de comunicações VoIP na empresa Wavecom, e algumas modificações na sua arquitectura surgiram a partir de desenvolvimentos provenientes desta empresa [152]. Essas diferenças são constituídas pelo modelo de gestão de informação em memória, ao invés de ficheiros XML guardados em disco, como configurações, *dialplans* e perfis SIP. Esta capacidade é adquirida por intermédio do módulo Lua denominado XML-Handler e o sistema memcached, com o objectivo de optimizar a performance do servidor. Foram aplicados testes extensos no Freeswitch não optimizado sobre a seguinte

configuração: CPU Intel Xeon 2.39 GHz, memória 2 GB, disco de 50 GB com 5400 rotações por minuto, sistema operativo Linux CentOS 6.4 64-bits, rede 1Gbps dedicados e Freeswitch versão 1.2.8 [153]. Através de *scripts* Lua para captar valores relativos ao servidor de telefonia no sistema operativo e a ferramenta VoIPmonitor com o intuito de registar métricas de qualidade de áudio, foram elaborados vários cenários de testes [154].

O primeiro cenário de testes centra-se na carga máxima de chamadas que o servidor VoIP suporta sem degradar a qualidade do áudio. O *codec* PCMA será uma referência nos testes devido à adesão por parte dos operadores actuais. Após cinco repetições dos testes com duração de dez minutos cada e trezentas chamadas PCMA geradas com intervalo de um segundo e duração aleatória também em cada repetição, os resultados são apresentados na Tabela 7.

Tabela 7 – Resultados do teste de carga máxima PCMA (Adaptada de [153]**)**

Estatísticas de Rede		
Carga de rede (Mbit/s)	Download Upload	45,2 45,2
Estatísticas da qualidade de áudio		
Jitter (ms)	7,15	
Delay (ms)	5021,4	
MOS	2,44	
Perda de pacotes (%)	1,93	
Utilização Sistema		
CPU (%)	99,7%	

Dado o volume de chamadas num curto espaço de tempo, em média novecentas e trinta e três, o processador chega praticamente à sua capacidade máxima e a perda de pacotes é de apenas 1,93%. No segundo cenário o teste de carga põe à prova os *codecs* principais, de forma a perceber qual era mais generoso com os recursos do servidor, sem recorrer a técnicas de *transcoding*. As diferenças em relação às condições do teste anterior verificam-se no aumento do número de repetições para quinze, no limite de ocupação do processador passado para 95%, de forma a não perder qualidade de áudio, e variação do número de chamadas a realizar. Os resultados obtidos são apresentados na Tabela 8 [153].

Tabela 8 – Resultados de testes de carga máxima para cada *codec* (Adaptada de [153])

Codec		PCMA	G722	Opus	Speex	iLBC
Max. Chamadas Simultâneas		275	285	255	290	310
Estatísticas de Rede						
Carga de rede (Mbit/s)	Download	41,08	41,08	23,15	15,12	14,34
	Upload	41,08	41,08	23,15	15,12	14,34
Estatísticas da qualidade de áudio						
Jitter (ms)		2,4	2,5	-	2,8	2,3
Delay (ms)		38,8	52,6	-	134	83,1
MOS		4,18	4,07	-	4,02	4,11
Perda de pacotes (%)		0%	0%	-	0%	0%
Utilização Sistema						
CPU (%)		86,5%	92,3 %	84,6%	87,7%	88,6%
Desvio Padrão CPU (%)		4,02%	3,57%	1,41%	1,98%	6,02%
Memória (%)		32,2%	33,7%	32,6%	29,5%	33,6%
Desvio Padrão Memória (%)		1,62%	1,45%	1,43%	1,43%	2,29%

É perceptível que o *codec* com melhor gestão de recursos é o iLBC, uma vez que permite criar mais chamadas, devido ao tamanho reduzido dos pacotes enviados para a rede. As estatísticas de qualidade de áudio do *codec* Opus não foram registadas pela inexistência de suporte ao mesmo por parte do VoIPmonitor [153].

Num terceiro panorama foram recolhidos dados sobre o *transcoding* entre *codecs* com as mesmas condições do segundo teste. Os valores são exibidos na Tabela 9.

Tabela 9 – Resultados do teste de carga máxima para *transcoding* de *codecs* (Adaptada de [153])

Codec	G722 <-> PCMA	Opus <-> PCMA	Speex <-> PCMA	iLBC <-> PCMA
Metodologia				
Max. Chamadas simultâneas	119	18	64	79
Estatísticas da qualidade de áudio				
Jitter (ms)	3,9	-	3,4	10,9
Delay (ms)	47,7	-	19,7	219,1
MOS	4,18	-	4,22	4,08
Perda de pacotes (%)	0%	0%	0%	0%
Utilização Sistema				
CPU (%)	87,6%	89,1%	85,8%	88,8%
Desvio Padrão CPU (%)	2,21%	3,35%	2,45%	2,51%
Memoria (%)	21,8%	18,6%	20,9%	17%
Desvio Padrão Memória (%)	0,33%	1,07%	0,12%	2,23%

Os dados são recíprocos para cada par de *codecs* e este processo é menos desgastante entre G.722 e PCMA. O último caso de testes avalia o desempenho das optimizações do Freeswitch mencionadas anteriormente ao utilizarem o *codec* PCMA, em que os *dialplans* são processados em memória com setenta chamadas e intervalo de criação das mesmas de quinhentos milissegundos, e dez segundos de duração de cada. Foram executadas quinze repetições, em que cada chamada teve dez minutos de longevidade. As estatísticas do cenário descrito são apresentadas na Tabela 10.

Tabela 10 – Testes de carga para optimizações do servidor (Adaptada de [153]**)**

Sistema	Default (PCMA)	Optimizado (PCMA)
Metodologia		
Max. Chamadas simultâneas	70	70
Utilização Sistema		
CPU (%)	69%	49%
Desvio Padrão CPU (%)	1,11%	0,98%
Memoria (%)	21%	24%
Desvio Padrão Memória (%)	0,85%	0,83%

Os esforços incluídos nas criações da Wavecom compensam com um ganho de 20% de CPU e uma perda de 3% de memória em relação à instalação por defeito do servidor IP--PBX.

Apesar do servidor destacado para esta dissertação não conter as componentes criadas pela Wavecom, estes testes, aliados aos oficiais fornecidos pelos criadores do Freeswitch, espelham a capacidade deste servidor em termos de desempenho e extensibilidade. Sabendo que a entidade VoIP em causa não efectua qualquer processamento de vídeo, não foram realizados testes adicionais [155]. Estes ensaios pesaram e revelaram-se decisivos para a escolha das plataformas *server-side* da aplicação criada por intermédio deste trabalho académico.

Testes de atraso

Ao desenvolver o *software* proveniente da proposta para esta dissertação, foi notória a demora desde que é formada uma chamada a partir do cliente WebRTC implementado e outro ponto SIP. Para averiguar a origem deste atraso, foram executadas capturas de

pacotes por intermédio do programa Wireshark [156]. Os elementos *hardware* que constituíram a base para estes ensaios estão expostos na Tabela 11.

Tabela 11 – Equipamentos de testes

	Características
Servidor	Máquina Virtual CPU Intel i5 com 4 núcleos a 2.50 GHz cada 2 GB de memória RAM Sistemas Debian 7 64-bit, Freeswitch e Node.js
Cliente	CPU Intel i5 com 4 núcleos a 2.50 GHz cada 8 GB de memória RAM Microsoft Windows 7 64-bit e Google Chrome

Os cenários específicos considerados foram as chamadas entre utilizadores registados na solução WebRTC criada e chamadas entre um *softphone* Blink e um utilizador da aplicação colaborativa final [157]. Os tempos de atrasos considerados são: tempo entre o pedido de uma chamada e o envio do respectivo pacote, tempo desde o primeiro pacote enviado pela origem de uma chamada até à notificação da mesma no destino, tempo desde o atendimento até à troca de multimédia e o tempo de términus de uma chamada. Os casos testados são compostos por ligações entre utilizadores WebRTC Experiment, chamada iniciada por intermédio da aplicação Blink para um utilizador do cliente mencionado e chamada originada através do mesmo cliente com destino a uma instância do *softphone* referido. Em cada um dos últimos cenários, foram produzidas dez sessões de cada vez para elaborar os limites relativos aos tempos de atraso.

Na experiência onde dois utilizadores WebRTC participam, verifica-se que em média passam entre dez e treze segundos até o cliente enviar o pedido de chamada ao servidor. Este tempo prende-se na recolha de candidatos ICE por parte da implementação WebRTC do navegador *web* e na formulação da mensagem SIP. Para reduzir este tempo, o conceito "Trickle ICE" terá de ser introduzido no Freeswitch [158]. Esta extensão do protocolo ICE consiste na aquisição de candidatos ICE ao longo de uma sessão, em vez de reuni-los numa lista no princípio de uma chamada [158]. Entre um a três segundos passados após a recepção do pacote inicial por parte do servidor, o utilizador a ser chamado recebe a

notificação da chamada, e em média seis segundos mais tarde, os dados multimédia são recebidos. Dado que o tempo efectivo de uma chamada que envolve utilizadores da aplicação WebRTC Experiment é em média vinte e quatro segundos, a maior parte desse tempo é gasto pelos respectivos *browsers* com o processamento das informações necessárias ao estabelecimento de uma sessão. O encerramento da sessão, em que os elementos visuais da interface relativos ao utilizador que saiu da conferência desaparecem, pode demorar entre dois segundos a um minuto neste caso em particular. A explicação para esta diferença de tempo reside nos tipos de multimédia envolvidos, em que uma sessão de áudio encerra prontamente e uma sessão com áudio e vídeo pode alongar até ao tempo máximo referido.

No caso em que a chamada é iniciada no cliente WebRTC para um *softphone* Blink, o atraso inicial observado na primeira experiência ao enviar o pacote da origem para o Freeswitch, assim como o tempo até ao aviso de nova chamada gerado no destino, ocorrem e desde que a chamada é aceite no *software* externo, a troca de multimédia é efectuada entre um a dois segundos. Ao terminar a chamada, se esta possuir vídeo, são precisos entre seis a trinta segundos se a acção for despoletada no Blink, caso contrário o findar da sessão é imediato.

O último cenário é validado através de chamadas originadas no Blink e o tempo entre o pedido de chamada até à notificação no cliente *web* abordado é de um segundo, logo os tempos considerados até esse limite são menores do que um segundo. Ao atender, os dados de áudio são trocados, em média, dez segundos após essa acção. Em relação ao terminar da sessão, se for efectuada pelo utilizador na aplicação *web* o tempo a concluir esta operação é de um segundo e se for no *softphone* então pode demorar até trinta e quatro segundos. Um dado importante verificado nestes testes é o facto do *codec* PCMA ser o eleito nas chamadas.

Testes de usabilidade

Para testar a usabilidade da versão beta da solução desenvolvida a partir desta dissertação, foi construído um documento, que se encontra no Anexo A, com tarefas e questões dirigidas a utilizadores que avaliaram a qualidade da interface, em relação às

funcionalidades do WebRTC Experiment. Seis provas foram realizadas e o tempo estimado para os utilizadores completarem o teste foi estabelecido em vinte minutos.

Todas as experiências respeitaram o limite de tempo proposto e revelaram-se produtivas para a melhoria da interface do WebRTC Experiment. Todas as tarefas propostas ocorreram com o grau de dificuldade mínimo nos seis testes, concluindo que a interface é intuitiva. As opiniões de quem testou sobre o visual do cliente demonstram que é agradável e os elementos mais relevantes estão bem destacados pelo esquema de cores adoptado. Outros reparos partilhados por mais do que um utilizador incluíram a atribuição de uma cor de fundo, adição de componentes visuais ou unificação das existentes, modificação do indicador de quando uma chamada está a ser efectuada e rapidez de carregamento de páginas. Em termos de comentários específicos, estes abrangem a inclusão de uma lista de contactos, reconhecimento visual do campo onde se introduz uma mensagem instantânea, chamadas de voz assinaladas como tal, descrição adicional do botão para desligar uma chamada, substituição de textos a negrito, remoção da borda do vídeo do utilizador e estabelecimento de uma sessão demorado. Um ponto negativo e não controlável pela aplicação é o pedido WebRTC para aceder à câmara e ao microfone, onde a maior parte dos utilizadores não conseguiu reparar de imediato, comprometendo o funcionamento esperado.

A partir da base formada através do *feedback* ganho com os exames efectuados, a interface sofreu alterações valiosas, tanto em pormenores como em acréscimo de funcionalidades, para concretizar um passo extra em direcção ao aperfeiçoamento do sistema aqui criado. Tais mudanças estão relatadas no ponto referente ao protótipo desse sistema.

6. Conclusão

A área das comunicações em tempo real terá sempre um lugar no mundo e a sua importância será crescente, seja pelo desenvolvimento de países do terceiro mundo ou por crises económicas que levam as empresas a realizar cortes de despesas. O estudo desta dissertação para conhecer as soluções actuais no âmbito da mesma, mostrou que as ofertas actuais são completas e robustas, mas ao mesmo tempo retêm algumas limitações ou revelam arquitecturas complexas. Confirmou-se uma certa inovação neste campo ao longo do tempo, mas a maior presenciada recentemente é o conceito WebRTC. Esta tecnologia traz um futuro auspicioso às comunicações em tempo real unificadas, ao promover extensibilidade, adaptabilidade e simplicidade. Apesar das suas qualidades, o desempenho observado em contextos distintos dita a necessidade de evolução, onde são notórios os esforços para fazer de WebRTC uma tecnologia de eleição.

Um dos objectivos desta dissertação consistia no desenvolvimento de um protótipo *web* capaz de suprimir certas limitações correntes, como a instalação de *software* externo no dispositivo do cliente, onde três ou mais pessoas podiam comunicar entre si e para isso foi criada a "WebRTC Experiment", que consegue reduzir a complexidade dos programas estudados através de tecnologias recentes, nomeadamente HTML5, WebSockets e Node.js, unificando os sistemas de comunicações em tempo real. A performance é um ganho adquirido com esta solução, como mostram os testes de cada componente central. As capacidades de criar sessões, trocar mensagens instantâneas entre os seus participantes, partilhar ficheiros e esboçar num *whiteboard* são cumpridas na plataforma proposta, sendo acessíveis por intermédio de uma interface adequada para qualquer utilizador.

Os requisitos do trabalho proposto foram assim concretizados, nomeadamente o estudo de tecnologias pertencentes à versão 5 do HTML e a concepção de uma solução *web* colaborativa com funcionalidades baseadas apenas no *browser* do utilizador.

6.1. Trabalho futuro

As dificuldades propensas a serem encontradas, devido ao estado inacabado dos *standards,* foram encontradas ao longo do desenvolvimento da aplicação resultante desta dissertação, com destaque para a compatibilidade da implementação WebRTC entre os navegadores *web*. Google Chrome é o *browser* de eleição para o cliente criado mas é necessário precaver as mudanças dos *standards* ainda em discussão. O Freeswitch também revelou incompatibilidades em relação ao suporte WebRTC, sendo ainda relativo.

Em termos de funcionalidades, a parte cliente do sistema aqui produzido tem as componentes básicas de um *software* colaborativo, excepto a partilha de ecrã. Esta encontra-se disponível na implementação WebRTC, mas não é suportada na biblioteca JsSIP até à data de entrega deste documento.

Algumas opções avançadas em falta relacionadas com uma sessão incluem o controlo local de áudio e vídeo de cada participante, e uma ferramenta em JavaScript capaz de converter ficheiros que contenham apresentações para imagens ou slides HTML5, para apresentar na interface do cliente. Acrescenta-se a possibilidade de integrar um modelo de login com credenciais na plataforma originada por intermédio desta dissertação, e o suporte para interligação de conferências, onde é facultada a escolha de ligação para participantes de outras conferências WebRTC Experiment e fundir duas ou mais das mesmas.

Referências

[1] Network Working Group, “Next Steps for the IP QoS Architecture,” 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2990.txt. [Accessed: 27-Oct-2013].

[2] W3Schools, “HTML5 Introduction,” 2013. [Online]. Available: http://www.w3schools.com/html/html5_intro.asp. [Accessed: 11-Jul-2013].

[3] G. Inc, “WebRTC,” 2012. [Online]. Available: http://www.webrtc.org/. [Accessed: 11-Jul-2013].

[4] Nature Publishing Group, “Public Television in Germany,” *Nature 137*, 1936. [Online]. Available: http://www.nature.com/nature/journal/v137/n3462/abs/137391a0.html. [Accessed: 18-Jul-2013].

[5] B. Goode, “Voice Over Internet Protocol (VoIP),” p. 1495;1517, 2002.

[6] W. REDMOND, “Microsoft Launches MSN Messenger Service,” 1999. [Online]. Available: http://www.microsoft.com/en-us/news/press/1999/jul99/messagingpr.aspx. [Accessed: 02-Aug-2013].

[7] SOFTONIC INTERNATIONAL S.L., “Windows live messenger,” 2013. [Online]. Available: http://en.softonic.com/s/windows-live-messenger. [Accessed: 02-Aug-2013].

[8] Net2Phone Inc, “Communication withuot borders,” 2011. [Online]. Available: http://web.net2phone.com/home_intpt.asp. [Accessed: 03-Aug-2013].

[9] C. Crouch, “MSN gives Messenger a voice,” *2000*. [Online]. Available: http://www.networkworld.com/news/2000/0720messenger.html. [Accessed: 03-Aug-2013].

[10] D. R. Flickinger, “WINDOWS® XP INTERACTIONS WITH UPnP™-BASED IGDs VERSION 1.0,” 2003. [Online]. Available: http://hometoys.com/emagazine.php?url=/htinews/aug03/articles/flickinger/upnp.htm. [Accessed: 06-Aug-2013].

[11] D. Feies, “MSN Messenger 7.5 Final (build 7.5.0299) just released,” 2005. [Online]. Available: http://blogs.msdn.com/b/dfeies/archive/2005/08/23/455268.aspx. [Accessed: 06-Aug-2013].

[12] R. Pelle, “Lançado Windows Live Messenger 8.1.0168.00 BETA,” 2006. [Online]. Available: http://www.hardware.com.br/comunidade/windows-live/672988/. [Accessed: 06-Aug-2013].

[13] K. Kniskern, “Messenger 9, GTalk integration, Messenger API, new client for Mac OS X – news unveiled at Georgia Tech presentation (whew),” 2007. [Online]. Available: http://www.liveside.net/2007/10/31/messenger-9-gtalk-integration-messenger-api-new-client-for-mac-os-x-news-unveiled-at-georgia-tech-presentation-whew/. [Accessed: 07-Aug-2013].

[14] P. Thurrott, “MSN Messenger 7.0 Review,” 2005. [Online]. Available: http://winsupersite.com/windows-live/msn-messenger-70-review. [Accessed: 07-Aug-2013].

[15] S. Pruitt, “MSN Messenger 6.0 Launched,” 2003. [Online]. Available: http://www.pcworld.com/article/111582/article.html. [Accessed: 07-Aug-2013].

[16] Damaster, “Wave 4: What’s New in Messenger?,” 2010. [Online]. Available: http://www.liveside.net/2010/04/28/wave-4-what-s-new-in-messenger/. [Accessed: 10-Aug-2013].

[17] Geekzone, “Microsoft Introduces Windows Live Messenger 8.0,” 2006. [Online]. Available: http://www.geekzone.co.nz/content.asp?contentid=6377. [Accessed: 10-Aug-2013].

[18] Sunshine, “Wave 3: Windows Live Messenger 9 Beta – What’s New? A Comparison With 8.5,” 2008. [Online]. Available: http://www.liveside.net/2008/09/17/wave-3-windows-live-messenger-9-beta-what-s-new-a-comparison-with-8-5/. [Accessed: 11-Aug-2013].

[19] Jonathan, “Save/import of contacts also gone in latest QFE3 update of Messenger 2009,” 2010. [Online]. Available: http://messengergeek.wordpress.com/2010/05/17/saveimport-of-contacts-also-gone-in-latest-qfe3-update-of-messenger-2009/. [Accessed: 11-Aug-2013].

[20] Microsoft, “Older Windows Live Technologies,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff747353.aspx. [Accessed: 05-Sep-2013].

[21] Microsoft, “Overview of the Windows Live Messenger Activity API,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa751014.aspx. [Accessed: 06-Sep-2013].

[22] Microsoft, “Windows Live Messenger Web Toolkit,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd570035.aspx. [Accessed: 07-Sep-2013].

[23] Microsoft, “Windows Live Messenger UI Controls,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd570061.aspx. [Accessed: 07-Sep-2013].

[24] Microsoft, “Windows Live Messenger Library,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/cc298458.aspx. [Accessed: 07-Sep-2013].

[25] Microsoft, “Windows Live ID SDK,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb404787.aspx. [Accessed: 07-Sep-2013].

[26] Microsoft, “Windows Live ID Delegated Authentication SDK for Application Providers,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/cc287637.aspx. [Accessed: 07-Sep-2013].

[27] Microsoft, “Windows Live ID Web Authentication SDK,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb676633.aspx. [Accessed: 07-Sep-2013].

[28] Microsoft, “What Is Messenger Connect?,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff749031.aspx. [Accessed: 08-Sep-2013].

[29] Microsoft, “About the Windows Live Admin Center SDK,” 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb259713.aspx. [Accessed: 08-Sep-2013].

[30] Microsoft, “O Messenger mudou para o Skype,” 2013. [Online]. Available: http://windows.microsoft.com/pt-pt/messenger/messenger-to-skype. [Accessed: 08-Sep-2013].

[31] Skype, “Videochamadas gratuitas, partilha de ecrã, chamadas através da internet, mensagens instantâneas e muito mais – Skype,” 2013. [Online]. Available: http://www.skype.com/pt/features/. [Accessed: 05-Jul-2013].

[32] Skype, “What platforms is Skype available on?,” 2013. [Online]. Available: https://support.skype.com/en/faq/FA12041/what-platforms-is-skype-available-on. [Accessed: 05-Jul-2013].

[33] Skype, “O que é o complemento do Skype?,” 2013. [Online]. Available: https://support.skype.com/pt/faq/FA12264/o-que-e-o-skype-plugin. [Accessed: 09-Sep-2013].

[34] Skype, “What is Skype Connect™ and how does it work?,” 2013. [Online]. Available: https://support.skype.com/en/faq/FA10549/what-is-skype-connect-and-how-does-it-work. [Accessed: 09-Sep-2013].

[35] S. Spillman, “What is GroupMe?,” 2011. [Online]. Available: http://help.groupme.com/entries/514793-What-is-GroupMe-. [Accessed: 10-Sep-2013].

[36] M. Jalali, “Skype 101 - Skype Architecture - Work in Progress,” 2009. [Online]. Available: http://www.mjalali.com/blog/?p=10. [Accessed: 12-Sep-2013].

[37] Skype, “What are P2P communications?,” 2013. [Online]. Available: https://support.skype.com/en/faq/FA10983/what-are-p2p-

communications?intcmp=blogs-_-generic-click-_-cio-update. [Accessed: 12-Sep-2013].

[38] Skype, “O Skype usa criptografia?,” 2013. [Online]. Available: https://support.skype.com/pt/faq/FA31/o-skype-usa-criptografia. [Accessed: 15-Sep-2013].

[39] Skype, “Skype and a New Audio Codec,” 2013. [Online]. Available: http://blogs.skype.com/2012/09/12/skype-and-a-new-audio-codec/. [Accessed: 20-Sep-2013].

[40] Skype, “CIO update: Post-mortem on the Skype outage,” 2013. [Online]. Available: http://blogs.skype.com/2010/12/29/cio-update/. [Accessed: 21-Sep-2013].

[41] M. Gillett, “What Does Skype’s Architecture Do?,” 2012. [Online]. Available: http://blogs.skype.com/2012/07/26/what-does-skypes-architecture-do/. [Accessed: 22-Sep-2013].

[42] BigBlueButton Inc, “History of BigBlueButton,” 2013. [Online]. Available: http://www.bigbluebutton.org/history/. [Accessed: 18-Sep-2013].

[43] BigBlueButton Inc, “Overview BigBlueButton,” 2013. [Online]. Available: http://bigbluebutton.org/overview/. [Accessed: 22-Sep-2013].

[44] BigBlueButton Inc, “Overview of BigBlueButton’s Architecture,” 2012. [Online]. Available: https://code.google.com/p/bigbluebutton/wiki/ArchitectureOverview. [Accessed: 24-Sep-2013].

[45] Adobe Systems Software Ireland Ltd, “Adobe Flash Player 11,” 2013. [Online]. Available: http://www.adobe.com/pt/products/flashplayer.html. [Accessed: 24-Sep-2013].

[46] A. Minessale and R. Chandler, “The World’s First Cross-Platform Scalable FREE Multi-Protocol Soft Switch.” [Online]. Available: http://www.freeswitch.org/. [Accessed: 25-Sep-2013].

[47] B. Fitzpatrick, “memcached - a distributed memory object caching system,” 2012. [Online]. Available: http://memcached.org/. [Accessed: 20-Sep-2013].

[48] I. Sysoev, “NGINX Plus for mission critical environments,” 2013. [Online]. Available: http://nginx.org/en/. [Accessed: 26-Sep-2013].

[49] C. Allen and J. Grden, “Red5 Media Server,” 2013. [Online]. Available: http://www.red5.org/. [Accessed: 26-Sep-2013].

[50] Citrusbyte, “Introduction to Redis.” [Online]. Available: http://redis.io/topics/introduction. [Accessed: 29-Sep-2013].

[51] GoPivotal Inc, “Learn Grails in 5 easy steps,” 2013. [Online]. Available: http://grails.org/learn. [Accessed: 30-Sep-2013].

[52] The Apache Software Foundation, “Apache Tomcat,” 2013. [Online]. Available: http://tomcat.apache.org/. [Accessed: 30-Sep-2013].

[53] Swftools Team, “SWF manipulation and generation utilities.” [Online]. Available: http://www.swftools.org/about.html. [Accessed: 01-Oct-2013].

[54] ImageMagick Studio LLC, “ImageMagick: Convert, Edit, And Compose Images,” 2013. [Online]. Available: http://www.imagemagick.org/script/index.php. [Accessed: 01-Oct-2013].

[55] Ghostscript Team, “Ghostscript.” [Online]. Available: http://www.ghostscript.com/. [Accessed: 01-Oct-2013].

[56] F. F. Dixon, “The DRAFT roadmap to BigBlueButton 1.0,” 2013. [Online]. Available: https://code.google.com/p/bigbluebutton/wiki/RoadMap1dot0. [Accessed: 02-Oct-2013].

[57] F. F. Dixon, “Overview an HTML5 client for BigBlueButton.” [Online]. Available: https://code.google.com/p/bigbluebutton/wiki/HTML5. [Accessed: 04-Oct-2013].

[58] Joyent Inc., “node.js,” 2013. [Online]. Available: http://nodejs.org/. [Accessed: 10-Oct-2012].

[59] The WebM Project, “About WebM,” 2013. [Online]. Available: http://www.webmproject.org/about/. [Accessed: 05-Oct-2013].

[60] J. Cerviño, P. Rodríguez, and Á. Alonso, “Web Real Time Communications Solutions.” [Online]. Available: http://lynckia.com/index.html. [Accessed: 03-Oct-2013].

[61] J. Cerviño, P. Rodríguez, and Á. Alonso, “Open Source WebRTC Communications Platform.” [Online]. Available: http://lynckia.com/licode/. [Accessed: 03-Oct-2013].

[62] J. Cerviño, P. Rodríguez, and Á. Alonso, “Overview of Licode and its components.” [Online]. Available: http://lynckia.com/licode/architecture.html. [Accessed: 04-Oct-2013].

[63] J. Cerviño, P. Rodríguez, and Á. Alonso, “Try it! Production applications to try Licode service.” [Online]. Available: http://lynckia.com/licode/try.html. [Accessed: 05-Oct-2013].

[64] Cisco, “Why WebEx? Connect with anyone, anywhere, any time,” 2013. [Online]. Available: http://www.webex.com/why-webex/overview.html. [Accessed: 31-Oct-2013].

[65] Cisco, “Compare WebEx products: Compare features across the WebEx product suite,” 2013. [Online]. Available: http://www.webex.com/products/compare-products.html. [Accessed: 31-Oct-2013].

[66] Cisco, “WebEx Mobile: WebEx goes with you!,” 2013. [Online]. Available: http://www.webex.com/products/web-conferencing/mobile.html. [Accessed: 31-Oct-2013].

[67] Cisco, “Cisco WebEx Web Conferencing, Online Meetings, Desktop Sharing, Video Conferencing,” 2013. [Online]. Available: http://www.webex.com/index.html. [Accessed: 02-Nov-2013].

[68] Cisco, “WebEx Meetings: Web Conferencing with high-definition video | WebEx,” 2013. [Online]. Available: http://www.webex.com/products/web-conferencing.html. [Accessed: 03-Nov-2013].

[69] Cisco, “Cisco WebEx Meetings - Product Overview,” 2013. [Online]. Available: http://www.wbximg.com/includes/documents/data-sheets/data_sheet_c78-707254.pdf. [Accessed: 03-Nov-2013].

[70] Cisco, “Cisco WebEx Network Bandwidth,” 2013. [Online]. Available: http://www.wbximg.com/includes/documents/data-sheets/Webex-Network-Bandwidth.pdf. [Accessed: 03-Nov-2013].

[71] Cisco, “WebEx Meetings Plans: Free, Premium, Premium Plus and Enterprise,” 2013. [Online]. Available: http://www.webex.com/plans/meetings-plans.html. [Accessed: 03-Nov-2013].

[72] Cisco, “eLearning, Online Training, Webinars, Desktop Sharing: Cisco WebEx Training Center,” 2013. [Online]. Available: http://www.webex.com/products/elearning-and-online-training.html . [Accessed: 03-Nov-2013].

[73] Cisco, “Cisco WebEx Event Center - Product Overview,” 2013. [Online]. Available: http://www.wbximg.com/includes/documents/data-sheets/ec_ds.pdf. [Accessed: 03-Nov-2013].

[74] Cisco, “Desktop Sharing, Online Support Software: Cisco WebEx Support Center,” 2013. [Online]. Available: http://www.webex.com/products/remote-support.html. [Accessed: 04-Nov-2013].

[75] Cisco, “Cisco WebEx Support Center Remote Support Product Overview,” 2013. [Online]. Available: http://www.wbximg.com/includes/documents/data-sheets/SCRS-Overview.pdf. [Accessed: 04-Nov-2013].

[76] Cisco, “Remote PC Access, Remote Support: Cisco WebEx Remote Access,” 2013. [Online]. Available: http://www.webex.com/products/remote-access.html. [Accessed: 04-Nov-2013].

[77] Cisco, “Cisco WebEx Support Center Remote Access Sell Sheet,” 2013. [Online]. Available: http://www.wbximg.com/includes/documents/data-sheets/ra_ds.pdf. [Accessed: 04-Nov-2013].

[78] World Wide Web Consortium, “HTML,” 2011. [Online]. Available: http://www.w3.org/community/webed/wiki/HTML. [Accessed: 08-Oct-2013].

[79] World Wide Web Consortium, “HTML5 Test Suite Conformance Results.” [Online]. Available: http://w3c-test.org/html/tests/reporting/report.htm. [Accessed: 08-Oct-2013].

[80] W3Schools, “HTML5 Web Storage,” 2013. [Online]. Available: http://www.w3schools.com/html/html5_webstorage.asp. [Accessed: 09-Oct-2013].

[81] W3Schools, “HTML5 Application Cache,” 2013. [Online]. Available: http://www.w3schools.com/html/html5_app_cache.asp. [Accessed: 09-Oct-2013].

[82] W3Schools, “HTML5 Geolocation,” 2013. [Online]. Available: http://www.w3schools.com/html/html5_geolocation.asp. [Accessed: 09-Oct-2013].

[83] W3Schools, “HTML5 Server-Sent Events,” 2013. [Online]. Available: http://www.w3schools.com/html/html5_serversentevents.asp. [Accessed: 09-Oct-2013].

[84] W3Schools, “HTML5 Drag and Drop,” 2013. [Online]. Available: http://www.w3schools.com/html/html5_draganddrop.asp. [Accessed: 09-Oct-2013].

[85] A. B. Johnston and D. C. Burnett, *WebRTC APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. Digital Codex LLC, 2012.

[86] Google Inc, “General Overview - WebRTC,” 2012. [Online]. Available: http://www.webrtc.org/reference/architecture.

[87] World Wide Web Consortium, “Participants in the Web Real-Time Communications Working Group - DBWG, the Working Groups Database,” 2013. [Online]. Available: http://www.w3.org/2000/09/dbwg/details?group=47318&public=1. [Accessed: 09-Jul-2013].

[88] World Wide Web Consortium, “Web Real-Time Communications Working Group,” 2013. [Online]. Available: http://www.w3.org/2011/04/webrtc/. [Accessed: 10-Oct-2013].

[89] Internet Engineering Task Force, “Search Internet-Drafts and RFCs,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/search/?name=SIP&activedrafts=on&rfcs=on. [Accessed: 21-Oct-2013].

[90] Internet Engineering Task Force, “Real-Time Communication in WEB-browsers (rtcweb),” 2013. [Online]. Available: http://datatracker.ietf.org/wg/rtcweb/. [Accessed: 10-Oct-2013].

[91] Network Working Group, “RFC 4566 - SDP: Session Description Protocol,” 2006. [Online]. Available: https://tools.ietf.org/html/rfc4566.

[92] Internet Engineering Task Force, “RFC 5245 - Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” 2010. [Online]. Available: http://tools.ietf.org/html/rfc5245.

[93] International Telecommunication Union, “G.711 : Pulse code modulation (PCM) of voice frequencies,” 2011. [Online]. Available: http://www.itu.int/rec/T-REC-G.711/en.

[94] International Communication Union, “G.722 : 7 kHz audio-coding within 64 kbit/s,” 2012. [Online]. Available: http://www.itu.int/rec/T-REC-G.722/e.

[95] Network Working Group, “RFC 3951 - Internet Low Bit Rate Codec (iLBC),” 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3951.txt.

[96] Network Working Group, “RTP Payload Format for the iSAC Codec,” 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-avt-rtp-isac-01.

[97] Network Working Group, “Definition of the Opus Audio Codec,” 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-codec-opus-16.

[98] Network Working Group, “RTP: A Transport Protocol for Real-Time Applications,” 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1889.txt.

[99] Network Working Group, “RFC 3605 - Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP),” 2003. [Online]. Available: http://tools.ietf.org/html/rfc3605.

[100] Network Working Group, “The Secure Real-time Transport Protocol (SRTP),” 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3711.txt.

[101] Network Working Group, “RFC 5389 - Session Traversal Utilities for NAT (STUN),” 2008. [Online]. Available: http://tools.ietf.org/html/rfc5389.

[102] BEHAVE Working Group, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” 2009. [Online]. Available: https://tools.ietf.org/html/draft-ietf-behave-turn-16.

[103] Team XSockets, “A fastlane to a plugin free video conference using WebRTC,” 2012. [Online]. Available: http://browsermeeting.com/.

[104] Google Inc, “Videochat demo, multiple users - Grupos do Google,” 2012. [Online]. Available: https://groups.google.com/forum/?fromgroups=#!topic/discuss-webrtc/RSOrjqWMsPo.

[105] Google Inc, “Google Chrome Frame,” 2013. [Online]. Available: http://www.google.com/chromeframe/?hl=pt-PT&quickenable=true.

[106] T. Levent, “How Will WebRTC Manifest Itself on Mobile Devices?,” 2012. [Online]. Available: http://bloggeek.me/webrtc-mobile/.

[107] Ericsson Labs, “Ericsson Labs | Bowser – The World’s First WebRTC-Enabled Mobile Browser,” 2012. [Online]. Available: https://labs.ericsson.com/blog/bowser-the-world-s-first-webrtc-enabled-mobile-browser.

[108] Network Working Group, “SIP: Session Initiation Protocol,” 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt. [Accessed: 11-Oct-2013].

[109] Google Inc, “webrtc - Web-based real-time communication.” [Online]. Available: https://code.google.com/p/webrtc/.

[110] Google Inc, “Datchannel info - Grupos do Google,” 2013. [Online]. Available: https://groups.google.com/forum/#!topic/discuss-webrtc/TT-0Q-njUiI.

[111] World Wide Web Consortium, “WebRTC 1.0: Real-time Communication Between Browsers,” 2013. [Online]. Available: http://dev.w3.org/2011/webrtc/editor/webrtc.html. [Accessed: 11-Oct-2013].

[112] World Wide Web Consortium, “Media Capture and Streams,” 2013. [Online]. Available: http://dev.w3.org/2011/webrtc/editor/getusermedia.html. [Accessed: 12-Oct-2013].

[113] W. W. W. Consurtium, “MediaStream Recording,” 2013. [Online]. Available: https://dvcs.w3.org/hg/dap/raw-file/tip/media-stream-capture/MediaRecorder.html. [Accessed: 11-Oct-2013].

[114] World Wide Web Consortium, “MediaStream Capture Scenarios,” 2013. [Online]. Available: https://dvcs.w3.org/hg/dap/raw-file/tip/media-stream-capture/scenarios.html. [Accessed: 12-Oct-2013].

[115] W. W. W. Consortium, “HTML Media Capture,” 2013. [Online]. Available: http://dev.w3.org/2009/dap/camera/. [Accessed: 13-Oct-2013].

[116] Internet Engineering Task Force, “RTCWeb Data Channels,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-data-channel/. [Accessed: 18-Oct-2013].

[117] Internet Engineering Task Force, “RTCWeb Data Channel Protocol,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-data-protocol/?include_text=1. [Accessed: 13-Oct-2013].

[118] Internet Engineering Task Force, “Javascript Session Establishment Protocol,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-jsep/?include_text=1. [Accessed: 16-Oct-2013].

[119] S. Dutton, “Getting Started With WebRTC - HTML5 Rocks,” 2013. [Online]. Available: http://www.html5rocks.com/en/tutorials/webrtc/basics/?redirect_from_locale=pt.

[120] Internet Engineering Task Force, “Overview: Real Time Protocols for Brower-based Applications,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-overview/?include_text=1. [Accessed: 15-Oct-2013].

[121] Internet Engineering Task Force, “Web Real-Time Communication (WebRTC): Media Transport and Use of RTP,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-rtp-usage/?include_text=1. [Accessed: 17-Oct-2013].

[122] Internet Engineering Task Force, “Security Considerations for WebRTC,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-security/?include_text=1. [Accessed: 19-Oct-2013].

[123] Internet Engineering Task Force, “WebRTC Security Architecture,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-security-arch/?include_text=1. [Accessed: 20-Oct-2013].

[124] Internet Engineering Task Force, “Transports for RTCWEB,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-transports/?include_text=1. [Accessed: 20-Oct-2013].

[125] Internet Engineering Task Force, “STUN Usage for Consent Freshness,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-stun-consent-freshness/?include_text=1. [Accessed: 21-Oct-2013].

[126] Internet Engineering Task Force, “Web Real-Time Communication Use-cases and Requirements,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-use-cases-and-requirements/?include_text=1. [Accessed: 20-Oct-2013].

[127] Internet Engineering Task Force, “WebRTC Audio Codec and Processing Requirements,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-rtcweb-audio/?include_text=1. [Accessed: 21-Oct-2013].

[128] D. Donohue, D. Mallory, and K. Salhoff, “Session Initiation Protocol,” 2006. [Online]. Available:

http://www.ciscopress.com/articles/article.asp?p=664148&seqNum=3. [Accessed: 21-Oct-2013].

[129] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Core,” 2011. [Online]. Available: http://xmpp.org/rfcs/rfc6120.html. [Accessed: 21-Oct-2013].

[130] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Address Format,” 2011. [Online]. Available: http://xmpp.org/rfcs/rfc6122.html. [Accessed: 21-Oct-2013].

[131] XMPP Standards Foundation, “XMPP Extensions.” [Online]. Available: http://xmpp.org/xmpp-protocols/xmpp-extensions/. [Accessed: 21-Oct-2013].

[132] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence,” 2011. [Online]. Available: http://xmpp.org/rfcs/rfc6121.html. [Accessed: 21-Oct-2013].

[133] Internet Engineering Task Force, “The WebSocket Protocol,” 2013. [Online]. Available: http://datatracker.ietf.org/doc/rfc6455/?include_text=1. [Accessed: 22-Oct-2013].

[134] World Wide Web Consortium, “The WebSocket API,” 2013. [Online]. Available: http://dev.w3.org/html5/websockets/. [Accessed: 21-Oct-2013].

[135] Google Inc, “Channel Python API Overview,” 2013. [Online]. Available: https://developers.google.com/appengine/docs/python/channel/. [Accessed: 22-Oct-2013].

[136] World Wide Web Consortium, “HTML5 Web Messaging,” 2012. [Online]. Available: http://www.w3.org/TR/webmessaging/#channel-messaging. [Accessed: 22-Oct-2013].

[137] Versatica, “JsSIP - the JavaScript SIP library.” [Online]. Available: http://www.jssip.net/. [Accessed: 15-Sep-2013].

[138] Versatica, “JsSIP - Interoperability,” 2013. [Online]. Available: http://www.jssip.net/documentation/misc/interoperability/. [Accessed: 15-Sep-2013].

[139] C. Engler, “englercj / node-esl - GitHub,” 2013. [Online]. Available: https://github.com/englercj/node-esl. [Accessed: 02-Sep-2013].

[140] A. Minessale and R. Chandler, “Event Socket Library,” 2013. [Online]. Available: http://wiki.freeswitch.org/wiki/Event_Socket_Library. [Accessed: 25-Oct-2013].

[141] Aaronontheweb, “Aaronontheweb | Intro to Node.JS for .NET Developers,” 2011. [Online]. Available: http://www.aaronstannard.com/post/2011/12/14/Intro-to-NodeJS-for-Net-Developers.aspx. [Accessed: 12-Apr-2013].

[142] M. Otto and J. Thornton, “Bootstrap,” 2013. [Online]. Available: http://getbootstrap.com/. [Accessed: 04-Sep-2013].

[143] S. Petre, “Colorpicker for Bootstrap, from Twitter,” 2013. [Online]. Available: http://www.eyecon.ro/bootstrap-colorpicker/. [Accessed: 16-Nov-2013].

[144] Admin, “Wallpaper Abstract - 1080p Wallpapers,” 2011. [Online]. Available: http://1920x1080.info/wallpaper-abstract/. [Accessed: 17-Sep-2013].

[145] “webrtcLogo,” 2013. [Online]. Available: https://apprtc-m.appspot.com/images/webrtcLogo.png. [Accessed: 16-Sep-2013].

[146] Flashlack, “File:Logo-webrtc.png,” 2013. [Online]. Available: http://commons.wikimedia.org/wiki/File:Logo-webrtc.png?uselang=pt. [Accessed: 16-Sep-2013].

[147] I. Novell, “OpenSUSE,” 2011. [Online]. Available: http://en.opensuse.org/Main_Page. [Accessed: 13-Nov-2013].

[148] UNH InterOperability Laboratory, “stunserver.org.” [Online]. Available: http://stunserver.org/. [Accessed: 20-Sep-2013].

[149] Network Working Group, “Session Initiation Protocol (SIP) Extension for Instant Messaging,” 2002. .

[150] R. F. Quixabeira, W. R. M. Almeida, and P. M. de A. Filho, *UTILIZAÇÃO DA TECNOLOGIA NODE JS PARA CONSUMO EFICIENTE DOS RECURSOS DE SERVIDORES WEB IN:Acta Brazilian Science Ano I, vol.01, 1^{o} trimestre*. São Luis, Ma, Brasil, 2013.

[151] chevymanjosh, “Long term testing chevymanjosh,” 2013. [Online]. Available: http://wiki.freeswitch.org/wiki/Long_term_testing_chevymanjosh. [Accessed: 24-Oct-2013].

[152] Wavecom, “Communications Engineering | Wavecom,” 2013. [Online]. Available: http://www.wavecom.pt/en/index.php. [Accessed: 20-Oct-2013].

[153] S. Reis, “Caracterização de uma Solução de Transcoding,” Faculdade de Ciências e Tecnologia - Universidade de Coimbra, 2013.

[154] VoIPmonitor, “VoIPmonitor - VoIP monitoring software - quality analyzer - WAV recorder,” 2012. [Online]. Available: http://www.voipmonitor.org/. [Accessed: 28-Oct-2013].

[155] A. Minessale, “Status of VP8 codec in Freeswitch,” 2012. [Online]. Available: http://permalink.gmane.org/gmane.comp.telephony.freeswitch.devel/5058. [Accessed: 05-Nov-2013].

[156] Wireshark Foundation, “Wireshark - Go Deep.,” 2013. [Online]. Available: http://www.wireshark.org/. [Accessed: 25-Nov-2013].

[157] AG Projects, “Blink, a state of the art, easy to use SIP client,” 2013. [Online]. Available: http://icanblink.com/. [Accessed: 11-Sep-2013].

[158] Internet Engineering Task Force, “Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol,” 2013. [Online]. Available: http://tools.ietf.org/html/draft-ivov-mmusic-trickle-ice-01. [Accessed: 16-Nov-2013].

Anexo A

WebRTC Experiment

Avaliação de usabilidade

Este documento constitui um conjunto de tarefas e questões com o propósito de avaliar a aplicação WebRTC Experiment participando duas pessoas, um utilizador e um observador. As tarefas listadas têm como objectivo estimar o comportamento do sistema perante as mesmas, sendo assim pedido que sejam realizadas descontraidamente.
O utilizador e o observador podem trocar perguntas durante esta avaliação. Adicionalmente, o utilizador tem a liberdade de, a qualquer momento, comentar determinado ponto, independentemente do contexto da questão ou tarefa em que se encontra. No final de cada tarefa pode ser requisitado um grau de dificuldade relacionado com a mesma.

1 – Muito Fácil; 5 – Muito difícil

Assim que estiver preparado informe o observador e inicie esta experiência. Execute o *browser* Google Chrome e aceda ao endereço indicado pela pessoa referida.

Questão 1: Quais são as suas primeiras impressões em relação à interface da aplicação?

_____

_____

Nota: _____

Tarefa 1: Faça login

1	2	3	4	5

Nota: _____

Questão 2: Quais são as suas primeiras impressões em relação à interface do cliente?

_____

_____

Nota: _____

Tarefa 2: Faça uma chamada para a Alice

1	2	3	4	5

Nota: _____

Tarefa 3: Realize uma chamada de vídeo para o Bob

1	2	3	4	5

Nota: _____

Tarefa 4: Desligue a chamada do Bob

1	2	3	4	5

Nota: _____

Tarefa 5: Efectue uma chamada para o número 501

1	2	3	4	5

Nota: _____

Tarefa 6: Faça novamente uma chamada para o Bob

1	2	3	4	5

Nota: _____

Tarefa 7: Envie uma mensagem instantânea à Alice

1	2	3	4	5

Nota: _____

Tarefa 8: Saia da aplicação

1	2	3	4	5

Nota: _____

Questão 3: Quais são os principais elementos visuais do cliente?

_____

_____

Nota: _____

Questão 4: Pode comentar a aplicação no geral?

_____

_____

_____

Nota: _____