UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE KREISMANN DOS SANTOS

Soluções *Open Source* para Interoperabilidade entre Sistemas de Videoconferência e Webconferência

Trabalho de Conclusão apresentado como
requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Prof. Dr. Valter Roesler
Orientador

Porto Alegre, fevereiro de 2017

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Kreismann dos Santos, Alexandre

Soluções *Open Source* para Interoperabilidade entre Sistemas de Videoconferência e Webconferência / Alexandre Kreismann dos Santos. – Porto Alegre: Graduação em Ciência da Computação da UFRGS, 2017.

70 f.: il.

Trabalho de Conclusão (bacharelado) – Universidade Federal do Rio Grande do Sul. Curso de Bacharelado em Ciência da Computação, Porto Alegre, BR–RS, 2017. Orientador: Valter Roesler.

1. Interoperabilidade. 2. Videoconferência. 3. Webconferência. 4. Open source. 5. Mconf. I. Roesler, Valter. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitor: Prof. Jane Fraga Tutikian
Pró-Reitor de Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Carla Maria Dal Sasso Freitas
Coordenador do Curso de Ciência da Computação: Prof. Sérgio Luis Cechin
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“Caminante, son tus huellas el camino, y nada más; caminante, no hay camino, se hace camino al andar. Al andar se hace camino, y al volver la vista atrás se ve la senda que nunca se ha de volver a pisar. Caminante, no hay camino, sino estelas en la mar.”

— Antonio Machado

AGRADECIMENTOS

Agradeço aos colegas mais próximos - vocês sabem quem são - que me acompanharam durante o curso. Nossos caminhos seguem estradas diferentes agora, porém minha gratidão seguirá sempre com vocês.

Também agradeço aos amigos e colegas do Mconf, sobretudo ao Mário Gasparoni Jr., sempre disposto e atencioso para me auxiliar.

Agradeço aos professores Taisy Weber, João Netto, Sérgio Cechin e Valter Roesler (meu orientador) não só pela compreensão e paciência, mas também pelas orientações e ações que me permitiram superar um momento difícil.

Finalmente, agradeço a minha mãe Sandra. Tu, sozinha, és mãe, pai, tio, tia, avô, avó; enfim, minha família inteira. Obrigado por ter sido tão forte. Te amo.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code
HTML5	Hypertext Markup Language, versão 5
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
ITU	International Telecommunication Union
MCU	Multipoint Control Unit
RFC	Request for Comments
RTP	Real-time Transport Protocol
RTMP	Real Time Messaging Protocol
SIP	Session Initiation Protocol
SDP	Session Description Protocol
URI	Uniform Resource Identifier
WebRTC	Web Real-Time Communication
XML	eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Sistemas de videoconferência permitem a comunicação por áudio e vídeo em tempo real e, devido a crescente qualidade das redes de banda larga, vêm apresentando um aumento significativo de uso, em que aulas, palestras, treinamentos e outros tipos de reuniões cada vez mais usufruem dessa tecnologia.

É possível dividir tais sistemas em quatro grandes grupos: *sistemas de webconferência*, que são executados em um navegador web; *sistemas desktop* que requerem a instalação em um computador; e, finalmente, os de hardware dedicado, que podem ser *sistemas de videoconferência de sala* ou *sistemas de telepresença*.

Embora cada grupo tenha o seu foco de uso, um dos fatores de qualidade de um sistema de videoconferência é a capacidade de interoperar com diferentes sistemas, pois há situações em que usuários de um sistema específico desejam se comunicar com utilizadores de outros softwares ou equipamentos. Entretanto, webconferências implementam, muitas vezes, diferentes padrões para sinalização, transmissão e codificação de áudio e vídeo, dificultando sua interoperabilidade com os demais tipos de videoconferência.

Este trabalho estudará os principais padrões e tecnologias de videoconferência e analisará soluções open source que os compatibilizem, permitindo que webconferências interoperem com os diversos sistemas de videoconferência. Usando tais soluções, será feito um estudo de caso com o sistema de webconferência open source Mconf, a fim de estender seu nível de interoperabilidade com demais softwares e equipamentos.

Palavras-chave: Interoperabilidade, videoconferenĉia, webconferenĉia, open source, Mconf.

Open Source Solutions for Interoperability of Video Conference and Web Conference Systems

ABSTRACT

Video conference systems allow real time communication through audio and video. Due to the rising quality of broadband networks, such systems are presenting a significant growth in use, in which classes, talks, trainings and other types of meetings are increasingly utilizing this technology.

We can divide such systems in four major groups: *web conferences*, which are executed in a web browser; *desktop systems* that require installation in a computer; and, finally, hardware based systems, which can be *room video conference systems* or *telepresence video conference systems*.

Although each type has its focus of use, a quality factor of a video conference system is the capability of interoperating with different systems, since there are situations in which a user of a specific system wishes to communicate with users operating another software or equipment. However, web conferences often implement different standards for media signaling, transmission and encoding, making more difficult their interoperability with further video conference types.

Thus the objective of this work is to study the main video conference standards and technologies and to analyze open source solutions that make them compatible, allowing the interoperability among web conferences and other video conference systems. Using these solutions, we're going to do a study case with the open source web conference system Mconf, extending its interoperability level with further softwares and equipments.

Keywords: interoperability, video conference, web conference, open source, Mconf.

1 INTRODUÇÃO

Sistemas de videoconferência possibilitam a comunicação de múltiplos usuários, independente de suas posições geográficas, através de áudio e vídeo. Com isso, aulas, treinamentos, apresentações e outros tipos de reunião podem ser realizados sem o deslocamento pessoal para um mesmo local físico, reduzindo custos e agilizando logísticas. Segundo (ROESLER et al., 2012), sistemas de videoconferência podem ser organizados em quatro grupos:

1. **Sistemas de videoconferência de sala**: São sistemas baseados em hardware e exigem o uso de salas físicas com um terminal de acesso - no caso, equipamentos de videoconferência acompanhados de televisões, monitores ou sistemas de som - como pode ser visto na Figura 1.1. Por serem *hardware based*, têm sua flexibilidade prejudicada, podendo depender de extensões Desktop para funcionalidades extras, tais como compartilhamento de tela ou apresentação de slides. Entretanto, justamente por serem hardwares dedicados, equipamentos de videoconferência de sala garantem áudio de alta qualidade e vídeo de alta definição em tempo real.

Figura 1.1: Exemplo de sistema de videoconferência de sala: equipamentos da Polycom.

Fonte: (ROESLER et al., 2012).

2. **Sistemas de videoconferência por telepresença**: são uma especialização dos sistemas de sala - herdando, portanto, as características descritas no item 1 - e têm como objetivo reproduzir a sensação de que todos os participantes estão na mesma sala física. Para tal, como mostra o exemplo da Figura 1.2, certas configurações de ambiente da sala e de equipamentos devem ser feitas, como:

    - Ajustar a câmera para mostrar o participante remoto em tamanho real;
    - Usar microfones e alto-falantes de uma forma que o som remoto venha a partir da posição do participante;

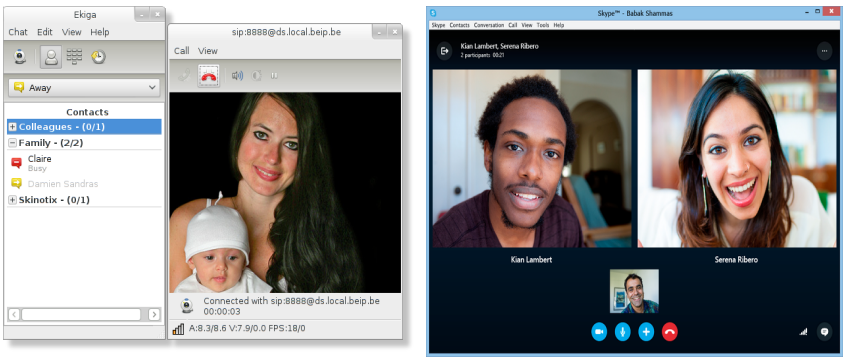
- Usar vídeo em alta definição para mostrar os detalhes dos participantes;
- Usar um ambiente complementar, com o mesmo tipo de cadeiras, cores e mesas.

Figura 1.2: Exemplo de sistema de telepresença da Cisco.

Fonte: (ROESLER et al., 2012).

3. **Sistemas de videoconferência desktop**: também conhecidos como softfones (junção das palavras *software* e telefone), são alternativas *software based* para sistemas de sala. Devem ser instalados em um computador pessoal, equipado com câmera e microfone. Embora tais sistemas não usem hardwares dedicados, a diferença de qualidade de mídia entre sistemas de sala e desktop tem diminuído, devido aos avanços na qualidade dos hardwares de PCs. A Figura 1.3 apresenta dois exemplos de softfones: Ekiga e Skype.

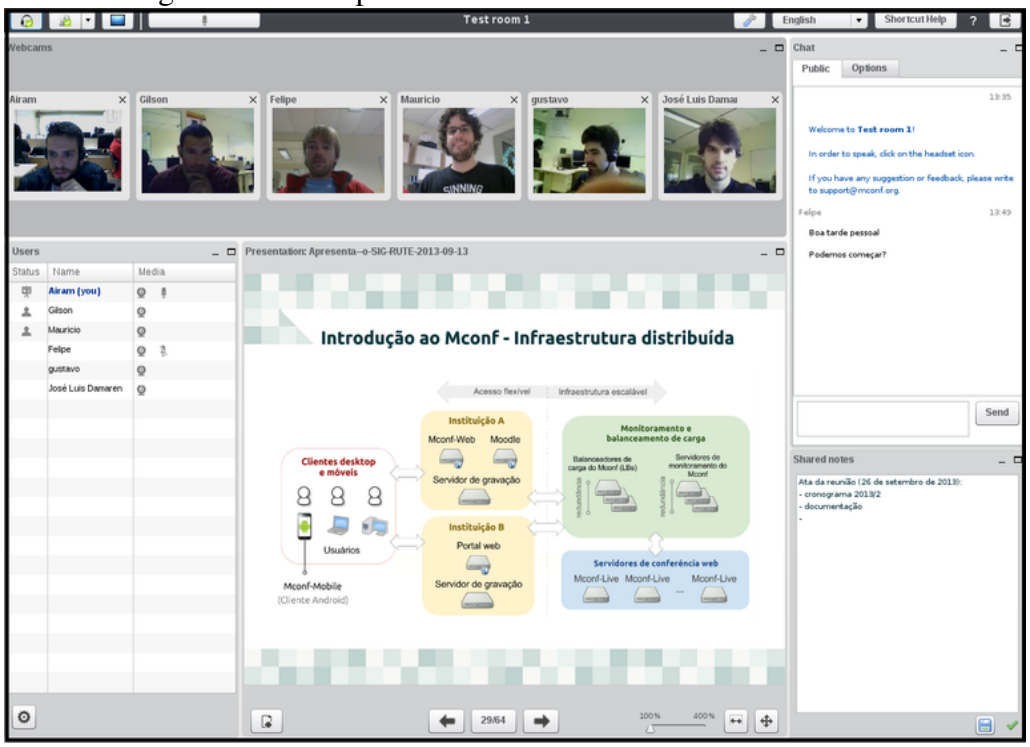
Figura 1.3: Exemplo de softfones: Ekiga (esquerda) e Skype (direita).



Fonte: (EKIGA, 2016); (SKYPE, 2016).

4. **Sistemas de videoconferência web** ou **sistemas de webconferência**: são sistemas de videoconferência executados em um navegador web, sendo compatíveis, portanto, com diferentes sistemas operacionais e não requerendo instalação. Para acessar um sistema de webconferência, o usuário, utilizando o sistema operacional de sua preferência, apenas precisa abrir uma URL em um bom navegador web, o que implica em uma grande facilidade de uso. Exemplos de tais sistemas são o Adobe Connect, Google Hangout e Mconf (este é demonstrado na Figura 1.4).

Figura 1.4: Exemplo de sistema de webconferência: Mconf.



Fonte: (MCONF, 2016).

1.1 Interoperabilidade

Por meio de padrões de videoconferência, um sistema desse tipo deve ser capaz de interoperar, através de áudio e vídeo, com diferentes sistemas (equipamentos ou softwares). Por exemplo, se um sistema utiliza os padrões SIP para sinalização e RTP para o transporte de mídia (itens que veremos mais adiante), espera-se que ele possa se comunicar com outro sistema que também os empregue.

Embora sistemas de videoconferência de sala e softfones trabalhem com um escopo similar de padrões (basicamente: SIP ou H.323, e RTP), sistemas de webconferência apresentam algumas particularidades. Por exemplo, se um sistema foi desenvolvido sobre a plataforma Flash, o transporte de mídia utilizará o padrão RTMP, em vez de RTP. Já webconferências utilizando-se da tecnologia WebRTC, que possibilita navegadores web utilizarem SIP e RTP, apresentam VP8, OPUS e outros padrões de codificação de mídia ainda não encontrados em muitos equipamentos de videoconferência e softfones.

Quando não há os mesmos padrões implementados, a interoperação fica dependente de soluções intermediárias que possibilitem a ponte entre esses padrões. Exemplificando, caso um equipamento de videoconferência utilize H.323 para sinalização e RTP para o transporte de mídia, e um sistema de webconferência empregue, respectivamente, SIP e RTMP, precisaremos de duas pontes para haver interoperabilidade: uma que viabilize a comunicação H.323/SIP e a outra, RTP/RTMP.

1.2 Objetivo

Dessa forma, considerando os quatro grupos de videoconferência, e o fato de que sistemas de webconferência, a depender da tecnologia que os compõe - Flash ou WebRTC -, muitas vezes implementam diferentes padrões de sinalização, transmissão e codificação

de áudio e vídeo, o objetivo deste trabalho é levantar e analisar soluções open source que permitam a ponte entre os diversos padrões existentes. Assim, ao utilizá-las, espera-se que sistemas de webconferência possam se comunicar de maneira transparente com os demais tipos de videoconferência.

Primeiramente, será feito o estudo dos principais e/ou atuais padrões e tecnologias empregados na construção de sistemas de videoconferência. Após, apresentação das soluções open source que compatibilizam os padrões estudados. Como veremos no decorrer deste trabalho, não há uma solução única que cubra todos os padrões, de forma que é preciso integrar diversas delas em uma arquitetura para abranger um número maior desses. Finalmente, será feito um estudo de caso com o sistema de webconferência open source Mconf, em que, através da integração das soluções apresentadas, se estenderá o nível de interoperabilidade da webconferência com demais softwares e equipamentos de videoconferência.

1.3 Organização

O trabalho está estruturado da seguinte forma:

- Capítulo 2: Apresentação de padrões de videoconferência e tecnologias: protocolos de sinalização (H.323, SIP), protocolos de transporte de mídia (RTP, RTMP), codificação de áudio (PCM, G.7xx, Speex, OPUS, iSAC, iLBC), codificação de vídeo (H.26x, VPx) e tecnologias para webconferência (Flash, WebRTC);
- Capítulo 3: Levantamento de soluções de interoperabilidade *open source*: Freeswitch, FFMPEG, Red5, Asterisk e Kurento;
- Capítulo 4: Integração das soluções de interoperabilidade, utilizando o sistema de webconferência Mconf como estudo de caso;
- Capítulo 5: Resultados da implementação proposta no Capítulo 4, demonstrando a interoperabilidade do sistema Mconf com diferentes sistemas de videoconferência;
- Capítulo 6: Considerações finais e sugestões para trabalhos futuros.

2 PADRÕES E TECNOLOGIAS DE VIDEOCONFERÊNCIA

Segundo (FIRESTONE; RAMALINGAM; FRY, 2007), as principais camadas de um sistema de conferência são:

- Interface de usuário e de administrador;
- Controle de Conferência: controle de criação/finalização de sessões, de roteamento de ligações e de alocações de recursos em geral;
- Plano de Controle: controle de chegada/saída de conexões e da negociação de parâmetros da sessão de acordo com a capacidade do servidor de mídia;
- Plano de Mídia: controle das *streams* de mídia, administrando o transporte, codificação/decodificação e a mixagem de áudio e vídeo.

O principal desafio para interoperabilidade entre sistemas de videoconferência é considerar os diferentes Planos de Controle e de Mídia existentes. São essas camadas que definem de que maneira um sistema envia e recebe requisições para entrada e saída de usuários, bem como o modo no qual ele irá transmitir e receber áudio e vídeo. Um sistema de conferência com Plano de Controle e de Mídia flexíveis permitem que usuários utilizando diferentes equipamentos ou softwares de videoconferência - os chamados *endpoints* - se conectem e se comuniquem de maneira transparente em uma sessão desse sistema.

Como afirmado em (KUROSE; ROSS, 2013), com a popularização das aplicações de tempo real, importantes órgãos, como o IETF e ITU, estabeleceram (e continuam estabelecendo) padrões para essa classe de aplicações. No contexto de videoconferência, padrões amplamente usados no Plano de Controle são os protocolos de sinalização SIP e H.323, os quais veremos na Seção 2.1. Já no Plano de Mídia, dependemos tanto de padrões de transporte como de codificação de mídia (Seções 2.2 e 2.3). Já em termos de webconferência, a tecnologia que compõe o sistema determinará o escopo de padrões usados, como será detalhado na Seção 2.4.

A seguir, portanto, são apresentados alguns dos principais padrões e tecnologias para sistemas de conferência.

2.1 Protocolos de Sinalização

Os protocolos de sinalização têm como papel a implementação do Plano de Controle. Para isso:

1. Determinam-se as portas onde o sistema irá aguardar conexões de *endpoints*. Tais portas definem o **canal de sinalização** de um sistema de conferência;

2. No momento em que um *endpoint* se conecta no canal de sinalização, o Plano de Controle envia informações da capacidade de mídia do sistema, as quais o *endpoint* deve respeitar. Esse passo estabelece a **negociação de mídia**;

3. Após essa negociação (bem-sucedida), o Plano de Controle e o *endpoint* abrem um canal lógico para *streaming* de mídia, e o *endpoint* passa a se comunicar, então, com o Plano de Mídia, para enviar e/ou receber áudio/vídeo. O Plano de Controle será responsável por administrar esse canal lógico, avisando o Plano de Mídia se o *endpoint* sair da sessão, ou algum erro de conexão ocorrer.

Como dito na introdução desse capítulo, existem dois principais padrões para a sinalização. O primeiro, H.323, foi estabelecido pelo ITU em 1996. O segundo, SIP, foi projetado pelo IETF e primeiramente definido no RFC 2543 em 1999.

2.1.1 Padrão H.323

Como afirmado em (TANENBAUM; WETHERALL, 2011), H.323, mais do que um protocolo específico, é uma visão geral da arquitetura de telefonia da Internet, bem como um encapsulador de diversos outros protocolos (que veremos no decorrer desse trabalho). O padrão, primeiramente, define quatro tipos de unidades (ilustrados na Figura 2.1):

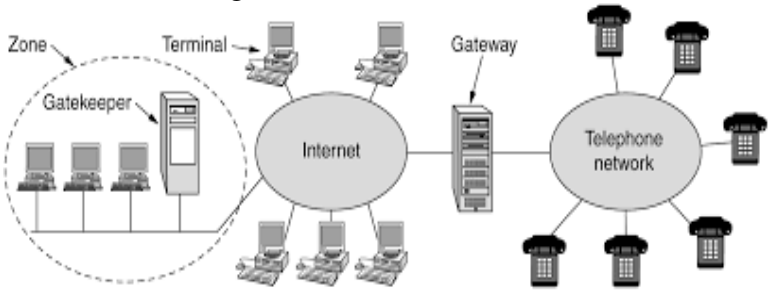
- **Terminais**: são os *endpoints*, emissores e receptores de mídia. É a única unidade obrigatória;

- ***Gatekeeper***: define a *zona* - conjunto de terminais que o gatekeeper irá administrar -, controlando quais *endpoints* podem se comunicar entre si, provendo acesso e controlando a largura de banda utilizada pelos terminais;

- ***Gateway***: permite a interoperação com terminais que utilizam redes de telefonia ou outros tipos de rede (que não a internet). Na prática, ainda permite a interoperabilidade com terminais que utilizem SIP, em vez de H.323, para sinalização;

- ***MCU (Multipoint Control Unit)***: sistema que permite que múltiplos terminais se conectem em uma sessão de conferência, distribuindo e mixando a mídia para os participantes.

Figura 2.1: Unidades H.323.

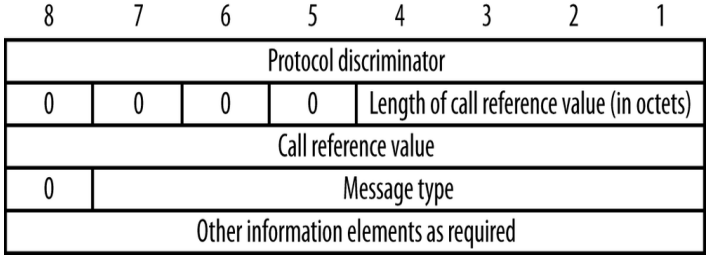


Fonte: (TANENBAUM; WETHERALL, 2011).

2.1.1.1 ***H.225***

O protocolo H.225 define o canal de sinalização de uma sessão H.323, descrevendo, portanto, como uma chamada é iniciada, estabelecida e finalizada. Um terminal iniciando uma chamada (daqui para frente chamado de *caller*), deve, primeiramente, abrir uma conexão TCP com a unidade que se quer chamar (respectivamente, *callee*), que estará esperando conexões na porta 1720. Um pacote H.225 é ilustrado na Figura 2.2 e é definido da seguinte forma:

Figura 2.2: Pacote H.225.

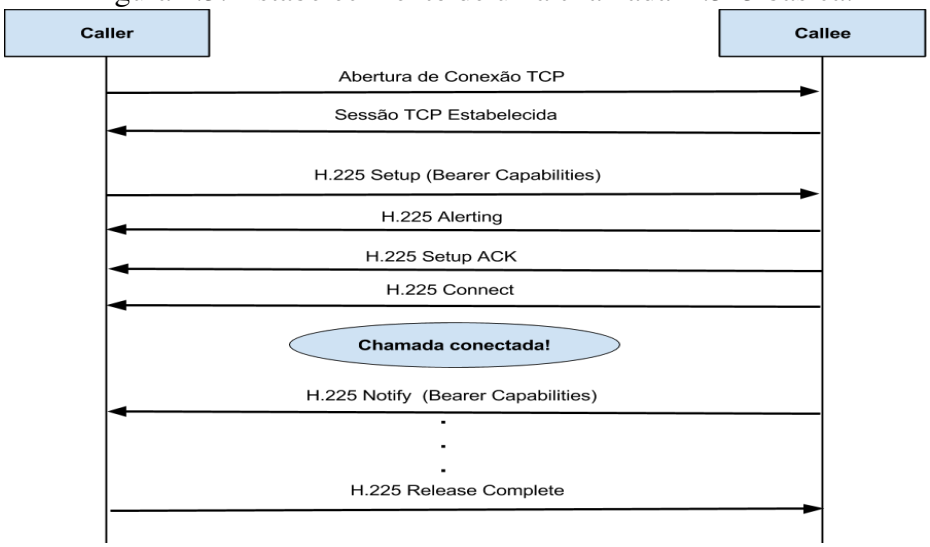


Fonte: (HARTPENCE, 2013).

- *Protocol Discriminator*: Número que identifica a mensagem como uma mensagem H.225. O valor desse campo é 8.
- *Length of Call Reference Value*: comprimento do CRV, que pode ser 1 ou 2 bytes;
- *Call Reference Value*: CRV é o número identificador único da chamada;
- *Message Type*: tipo da mensagem que o pacote H.225 carrega (*Setup Message, Connect, Alerting*, etc - definidos mais adiante).
- *Information Elements*: espaço para parâmetros adicionais. Dependem do *Message Type*.

A Figura 2.3 demonstra o estabelecimento de uma chamada H.323 básica:

Figura 2.3: Estabelecimento de uma chamada H.323 básica.



1. Após o estabelecimento da conexão TCP, o caller inicia a chamada, mandando um H.225 *Setup Message*. Esse tipo de *Message Type* carrega como parâmetro obrigatório o *Bearer Capability*, que define a natureza da chamada: se é apenas de áudio ou uma chamada audiovisual.

2. O *callee* responde o *Setup Message*, com um H.225 *Alerting*, indicando ao *caller* que o dispositivo chamado já iniciou o procedimento de alerta de que uma nova chamada chegou - em outras palavras, que o dispositivo está 'tocando' (como um telefone; em inglês, *Ringing*). Antes do *Alerting*, outro tipo de *Message Type* pode ocorrer (não é obrigatório) do *callee* para o *caller*: o *Call Proceeding*. CP tem o objetivo de informar o *caller* que a chamada está em processo de ser estabelecida.

3. O *callee* também responde um *Setup ACK*, indicando que o *Setup Message* foi devidamente recebido.

4. Quando a chamada é finalmente atendida, é mandado um H.225 *Connect* para o *caller*. É nesse momento que a conexão H.225 foi completamente estabelecida.

5. Mensagens H.225 *Notify* servem para troca de informações gerais entre *caller* e *callee* durante a chamada. Um uso é para o *callee* indicar ao *caller* o seu próprio *Bearer Capability*, como indicado na Figura 2.3.

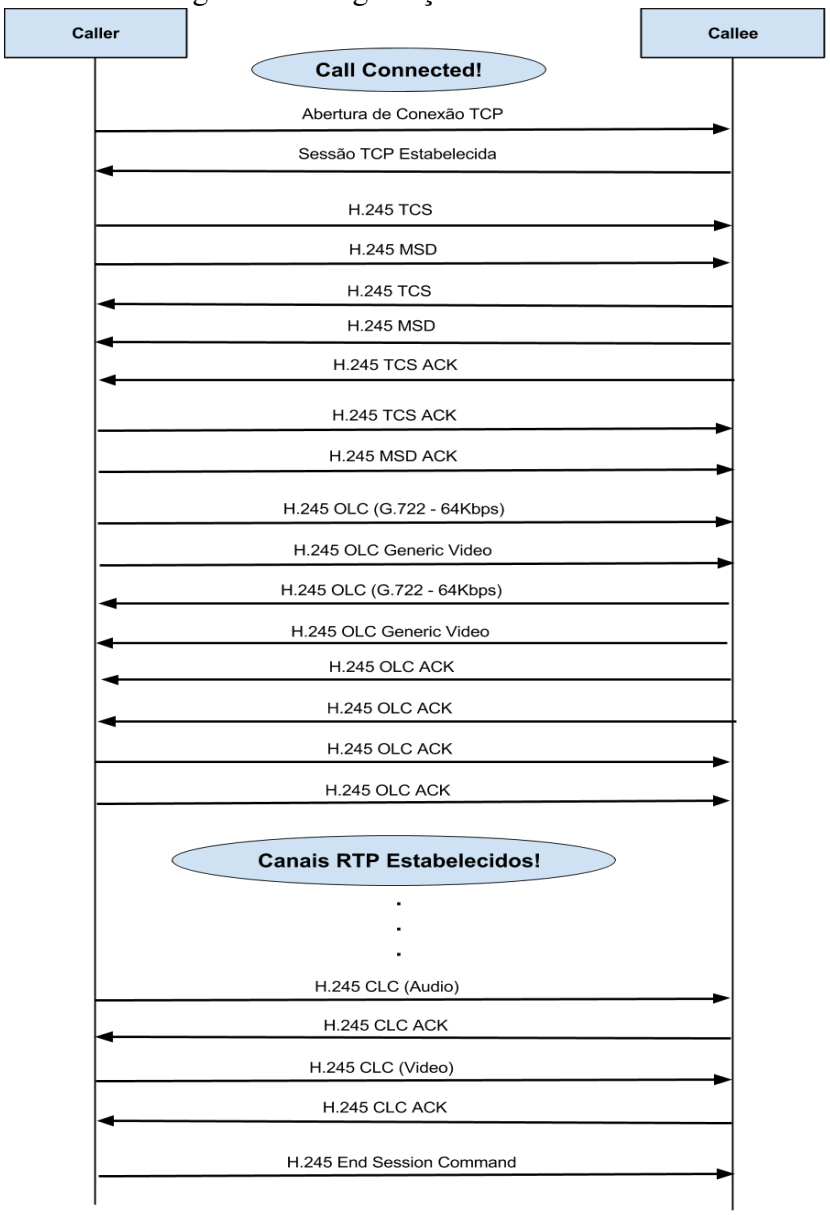
6. Quando uma das partes envolvidas encerra a chamada, é enviado um H.225 *Release Complete*. Com isso, o CRV, identificador único da chamada, é desalocado - assim como quaisquer outros recursos alocados - e pode ser reusado em outra chamada. O H.225 *Release Complete* carrega consigo um código que determina qual foi o motivo do encerramento da chamada (a Recomendação ITU-T H.225.0 (ITU, 2009a) contém uma lista de todos os códigos de encerramento e seus significados).

2.1.1.2 ***H.245***

O protocolo H.245 provê o mecanismo para a negociação de mídia e para a abertura dos canais RTP, onde os dados multimídia serão transmitidos/recebidos. Após o H.225 *Setup Message*, o *callee* pode usar o H.225 *Call Proceeding*, *Alerting* ou *Connect* para enviar ao *called* o IP e a porta para a abertura da sessão H.245. Quando o *caller* recebe essas informações, é aberta uma conexão TCP entre as partes envolvidas, onde haverá, então, a negociação de mídia.

A Figura 2.4 demonstra como é feita a negociação de mídia via sessão H.245.

Figura 2.4: Negociação de mídia H.323.



1. Após o estabelecimento da chamada - como visto anteriormente na Seção 2.1.1.1 - e da conexão TCP para a Sessão H.245, *caller* e *callee* trocam, primeiramente, 2 tipos de mensagens H.245: *TCS* (*Terminal Capability Set*) e *MSD* (*Master-Slave Determination*).

2. *TCS* contém a descrição da capacidade de mídia do remetente da mensagem. Inclui a lista de todos os codecs de áudio e vídeo suportados e suas características (tipos de payload, perfis e resoluções de vídeo, taxa de bits e de frames, etc - itens que veremos nas próximas seções). Também contém o *Simultaneous Capability Set*, que define quais grupos de codecs podem ser usados simultaneamente. A parte que recebe o H.245 *TCS* informa ao transmissor da mensagem o recebimento da mesma com um *TCS ACK*.

3. A troca de mensagens *MSD* define quem irá ser o 'mestre' e o 'escravo' da sessão. O mestre será responsável por atribuir um ID para o canal lógico, gerar uma

chave quando há encriptação de mídia, entre outras tarefas de administração do canal lógico. Uma mensagem *MSD* contém um *Terminal Type* (um número) e um *Determination Number* (número aleatório). Cada parte envolvida compara os números recebidos na mensagem *MSD* com os seus próprios números e quem tiver o maior *Terminal Type* será o mestre. Em caso de empate, o mestre será quem tiver o maior *Determination Number*. Caso haja novamente um empate, ocorre uma nova negociação, com novas mensagens *MSD*.

4. Posteriormente à determinação do mestre e escravo, o *caller* envia o H.245 *OpenLogicalChannel* (*OLC*), requisitando a abertura do canal lógico. O canal lógico, no contexto H.323, representa o caminho usado para a transmissão de mídia. A mensagem *OLC* carrega consigo o codec de áudio e de vídeo que o *caller* usará na transmissão, bem como as características de cada codec (como explicado no item 2). Outros dados importantes do H.245 *OLC* são as portas RTCP (mais detalhes na Seção 2.2.1) que serão usadas no contexto RTP e o *LCN* (*Logical Channel Number*), identificador único do canal lógico. No exemplo da Figura 2.4, tanto o *caller* quanto o *callee* enviam um *OLC* requisitando um canal para transmitir áudio usando o codec G.722, e um outro *OLC* requisitando um canal para transmitir vídeo H.264. *Generic Video Capability*, transportada na mensagem *OLC*, é uma estrutura H.245 específica para informar características do codec H.264. Existem, ainda, outras mensagens H.245 específicas para controle de vídeo, como o *Flow Control*, usado para reajustar a taxa de bits máxima, e o *Miscellaneous Command*, utilizado para pausar, congelar, avançar a stream de vídeo, bem como fazer outros ajustes, como, por exemplo, alterar a resolução ou a taxa de frames.

5. Quem recebe o *OLC* deve examinar os detalhes do pedido de abertura do canal lógico, alocar os recursos necessários e responder com um H.245 *OLC ACK*, contendo os IPs e as portas RTP e RTCP que serão usados para receber a mídia.

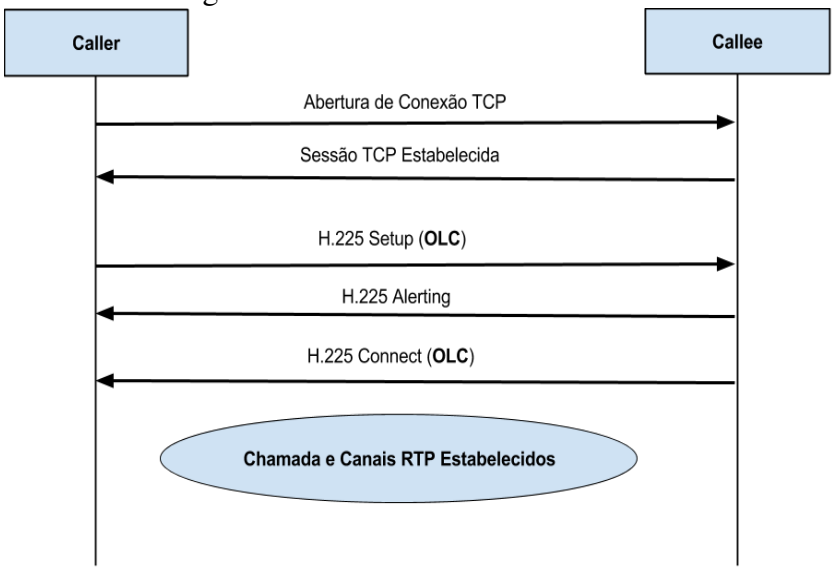
6. Quando uma das partes envolvidas encerra a chamada, é enviado um H.245 *Close Logical Channel* (*CLC*), que deverá ser respondido com um *CLC ACK*.

7. H.245 *End Session* é a mensagem que indica o fim da sessão. Após essa mensagem, não há mais trocas de nenhuma outra mensagem H.245 (portanto, não há ACK para *End Session*).

2.1.1.3 ***H.323 Fast Connect Mode***

No H.323 versão 2 existe a opção do *Fast Connect Mode* (Figura 2.5). O *caller* manda um H.245 *OLC* com um codec sugerido dentro do H.225 *Setup Message*. Se o *callee* suportar *Fast Connect Mode* e aceitar a sugestão do codec, será respondido um H.225 *Connect* também transportando um H.245 *OLC*. Com isso, as partes envolvidas já têm as informações necessárias para estabelecer a conexão de mídia, diminuindo significantemente o número de mensagens requeridas para estabelecer a sessão H.323. Segundo (FIRESTONE; RAMALINGAM; FRY, 2007), equipamentos de vídeo H.323 geralmente não suportam *Fast Connect Mode*.

Figura 2.5: H.323 *Fast Connect Mode*.



*2.1.1.4 **Mais informações sobre H.323***

Para mais informações sobre o protocolo H.323, consulte (ITU, 2009a), (ITU, 2009b), (ITU, 2011) e também (FIRESTONE; RAMALINGAM; FRY, 2007).

2.1.2 Padrão SIP

Segundo (TANENBAUM; WETHERALL, 2011), muitos viam o H.323 como um típico produto de uma companhia de telecomunicação: grande, complexo e inflexível. Nessa situação, o IETF projeta o padrão SIP, que, em vez de uma suíte de protocolos, é um módulo único. O protocolo SIP é *text-based*, apresentando todas as suas mensagens, atributos e métodos em ASCII. O padrão, primeiramente, define os seguintes elementos:

- ***User Agent***: podem ser tanto os *endpoints*, emissores e receptores de mídia, quanto servidores de controle de chamada (abordados nos próximos itens). Todo *UA* é formado por um *User Agent Client* (*UAC*) - responsável por iniciar as transações (que também definiremos mais adiante)- e por um *User Agent Server* (*UAS*) - responsável pelas respostas dessas transações;

- ***Proxy Server***: Entidade que faz o roteamento de mensagens SIP. É responsável por determinar para onde mensagens SIP vão ser direcionadas (quais *endpoints* ou servidores receberão determinada mensagem), provendo também autenticação e autorização de mensagens. Um PS *stateful* armazena o histórico de todas as mensagens de uma transação, a fim de levantar dados para distribuição de sessões SIP em diferentes servidores e/ou para geração de estatísticas. Já um PS *stateless* faz o roteamento de mensagens sem guardar quaisquer informações.

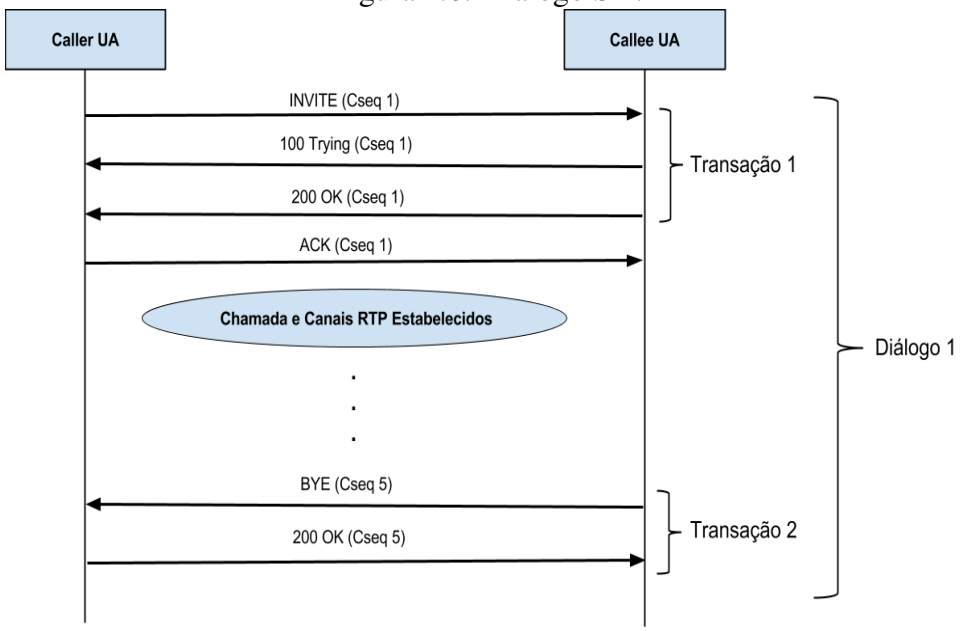
- ***Redirect Server***: Entidade que informa UAs para onde uma mensagem SIP deve ser encaminha. Um UA manda uma mensagem SIP, que deve ser entregue a outro UA, para o *Redirect Server*. Este, então, retorna para o UA de origem o endereço alternativo do UA destino. Em posse da nova informação, o UA remetente remanda a mensagem SIP diretamente para o UA destino. Portanto, não é o *Redirect Server* que faz o roteamento da mensagem (como o *Proxy Sever* faria): o trabalho de fazer

a nova ligação - utilizando o endereço certo advindo da resposta do RS - é do UA de origem.

- ***Registrar Server***: Entidade para onde os usuários enviam o pedido de registro. Usuários são registrados como URIs em um banco de dados, onde cada URI é associado com uma localização (normalmente o endereço IP desse usuário). Com isso, *Proxy Servers* ou *Redirect Servers* podem usar essa base de dados para executar suas tarefas.

Na Figura 2.6 abaixo, é demonstrado um diálogo SIP:

Figura 2.6: Diálogo SIP.



1. Começamos com o UA callee recebendo um *INVITE* do UA Caller, para o início de uma chamada. Tipicamente, a porta que define o canal de sinalização SIP é a porta 5060 (podemos encontrar ambientes onde se usa a porta 5070 ou 5061 - esta última para sinalizações encriptadas). Um *INVITE* tem a seguinte estrutura (Figura 2.7):

Figura 2.7: SIP Invite.

```
INVITE sip:UAcallee@172.27.14.53 SIP/2.0

! As linhas abaixo são os headers SIP:
Via: SIP/2.0/UDP 172.27.14.4:5060;branch=8dJXAX9MDw
Max-Forwards: 70
To: <sip:UAcallee@172.27.14.53>
From: <sip:UAcaller@172.27.14.4>;tag=ds17aa9bd4
Call-ID: 11022705439144@172.27.14.4
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Content-Length: 251
Content-Type: application/sdp

! Já as linhas abaixo estão no body do INVITE SIP:
v=0
o=UACaller 1549546120 0 IN IP4 172.27.14.4
s=ExampleSession
c=IN IP4 172.27.14.4
t=0 0
m=audio 49220 RTP/AVP 118
a=rtpmap:118 Speex/16000/1
m=video 5068 RTP/AVP 104
a=rtpmap:104 H264/90000
a=fmtp:104 max-fs=6336;max-mbps=190080;profile-level-id=42801e
```

O INVITE é acompanhado do endereço de destino (no caso, o usuário 'UAcalle' que está no endereço IP 172.27.14.53) e tem os seguintes cabeçalhos:

(a) *Via*: indica como transportar a mensagem (no caso, usando UDP) e para onde as respostas dessa mensagem devem ir (IP e porta de destino). Ainda contém o *branch*, identificador único da transação SIP. Uma transação, no contexto SIP, significa um *Request* - requisição - seguido de suas *Responses* - respostas (pode ser visto na Figura 2.6 que o ACK não está incluído na Transação 1, isso pois, em geral, ACK não é considerado parte de transações). Já um diálogo SIP é o conjunto de todas as transações entre 2 UAs. Nas Tabelas 2.1 e 2.2 são definidos os SIP *Requests* e os SIP *Responses* mais comuns segundo (FIRESTONE; RAMALINGAM; FRY, 2007);

(b) *Max-Forwards*: usado para evitar *loops*. Toda vez que um *Request* é redirecionado, o *proxy* que recebe esse *Request* decrementa o *Max-Forwards* em 1. Se um *proxy* recebe um *Request* com *Max-Forwards* igual a 0, ele responde com uma mensagem de erro para o UA de origem;

(c) *To*: identificador do destino: usuário seguido do seu endereço IP;

(d) *From*: identificador da origem: usuário seguido do seu endereço IP. O parâmetro *tag* é usado para identificar o diálogo SIP;

(e) *Call-ID*: identificador único da chamada SIP;

(f) *Cseq*: o *Command Sequence* é um identificador de transação, sendo um número (que pode ser arbitrário) acompanhado do nome da *Request* que originou a transação. Todas as *Requests* e *Responses* de uma transação devem usar o mesmo *Cseq*;

(g) *Allow*: lista de *Requests* que o UA pode enviar e receber;

(h) *Content-Length*: número de caracteres do *body*;

(i) *Content-Type*: do que consiste o body; no caso, do SDP.

Tabela 2.1: SIP Requests.

SIP Request	**Descrição**
INVITE	Convida um UA para uma chamada.
BYE	Termina o diálogo SIP entre 2 UAs.
OPTIONS	Solicita informações das capacidades do UA remoto.
MESSAGE	Manda mensagens instantâneas (não faz parte de um diálogo).
ACK	Confirma que o UA *caller* recebeu uma *Response* definitiva (que não é da classe *1xx*) sobre o seu pedido de INVITE.
REGISTER	Solicitação para ser registrado em um servidor de registro.
CANCEL	Cancela um *Request* corrente.
INFO	Método para UAs trocarem informações gerais durante uma sessão SIP.
PRACK	Age como um ACK temporário, onde um UA confirma que recebeu alguma *Response* provisória (da classe *1xx*)
UPDATE	Atualiza o estado da sessão SIP.
SUBSCRIBE	Solicita inscrição para receber notificações da ocorrência de determinado(s) evento(s) SIP.
NOTIFY	Envia um evento de notificação para UAs inscritos.
REFER	Indica que um UA deve usar o endereço contido no REFER para refazer um determinado *Request*.

Tabela 2.2: SIP Responses.

SIP Response Code	**Reason**	**Response Class**
100	Trying	1xx
180	Ringing	1xx
200	OK	2xx
301	Moved permanently	3xx
302	Moved temporarily	3xx
400	Bad Request	4xx
600	Busy	6xx
603	Decline	6xx
604	Does not exist	6xx

O SDP (*Session Description Protocol*) define a sintaxe da descrição da sessão SIP de mídia. É através da troca de SDPs entre origem e destino que é feita a negociação de mídia. No exemplo da Figura 2.7, o SDP contém os seguintes parâmetros:

(a) *v*: versão do SDP;

(b) *o*: identificador do criador da sessão;

(c) *s*: nome da sessão;

(d) *c*: informação da conexão. Basicamente, de qual IP virá a mídia. Esse é um exemplo de atributo que pode ser tanto:

- *session-level*: vale para toda a sessão. Para isso, o atributo deve estar listado antes da descrição das mídias. No exemplo da Figura 2.7, o parâmetro *"c="* está justamente como atributo de sessão, ou seja, todas as mídias que serão descritas posteriormente virão do mesmo IP;

- *media-level*: vale apenas para uma mídia específica. Para isso, o atributo deve estar listado depois de uma descrição de mídia (*"m="*). Com isso, por exemplo, poderíamos declarar diferentes informações da conexão para diferentes mídias (assim, áudio e vídeo poderiam vir de IPs distintos).

(e) *t*: define o tempo de início e o tempo de fim da sessão SIP. Os valores de *t* são representações decimais dos valores de tempo do *Network Time Protocol* (NTP), que, por sua vez, são os valores de tempo em segundos desde 1900. No exemplo da Figura 2.7, ambos os valores são zero, significando que o começo da sessão é imediato (assim que negociação de mídia terminar) e sem tempo de término (sessão só será encerrada quando uma das partes envolvidas desligar a chamada);

(f) *m*: descreve as mídias que serão usadas. No nosso exemplo, tanto áudio, quanto vídeo. A linha *"m="* ainda define a porta de destino da mídia (49220 para áudio e 5068 para vídeo), o protocolo de transporte usado na transmissão da mídia (RTP para ambos) e a lista de *payload types* (Seção 2.2.1) de cada codec de mídia;

(g) *a*: descrição de um atributo. Atributos podem ser *session-level* (atributos de sessão) ou *media-level* (atributos de mídia). Um uso importante desse parâmetro é descrever os codecs de mídia que um UA suporta, logo após a descrição de mídia (*"m="*). Na Figura 2.7, é estabelecido que o UACaller só suporta Speex como codec de áudio e H.264 como codec de vídeo. Ao final do SDP, há ainda alguns atributos específicos do codec H.264, como tamanho máximo de frame, taxa máxima de bits por segundo e *profile* de vídeo (ver Seção 2.3). Vale a menção que há diversos outros usos para atributos e existe uma lista extensa definida em (IETF, 2002).

2. Quando o *callee* recebe a chamada e entra em processo para determinar se a aceitará ou não, é mandado um 100 Trying para o *caller* (Figura 2.8):

Figura 2.8: SIP 100 Trying.

```
SIP/2.0 100 Trying

Via: SIP/2.0/UDP 172.27.14.53:5060;branch=8dJXAX9MDw
To: <sip:UAcaller@172.27.14.4>
From: <sip:UAcallee@172.27.14.53>;tag=ds17aa9bd4
Call-ID: 11022705439144@172.27.14.4
CSeq: 1 INVITE
Content-Length: 0
```

3. Assim que o *callee* aceita a chamada, este manda um 200 OK para o *caller* (Figura 2.9). Nele, há o SDP descrevendo quais codecs de mídia, do conjunto que o *caller* enviou no SDP do INVITE, que o *callee* também suporta (no caso, o *callee* também suporta Speex e H.264). Há também, como importante atributo de mídia (tanto para áudio, quanto para vídeo), o IP e as portas RTCP que o *callee* usará para o controle do recebimento dos fluxos RTP (áudio e vídeo).

Figura 2.9: SIP 200 OK.

```
SIP/2.0 200 OK

Via: SIP/2.0/UDP 172.27.14.53:5060;branch=8dJXAX9MDw
To: <sip:UAcaller@172.27.14.4>
From: <sip:UAcallee@172.27.14.53>;tag=ds17aa9bd4
Call-ID: 11022705439144@172.27.14.4
CSeq: 1 INVITE
Content-Length: 317
Content-Type: application/sdp

v=0
o=UACaller 1549568353 0 IN IP4 172.27.14.53
s=ExampleSession
c=IN IP4 172.27.14.53
t=0 0
m=audio 31818 RTP/AVP 118
a=rtpmap:118 Speex/16000
a=rtcp:31819 IN IP4 172.27.14.53
m=video 29240 RTP/AVP 104
a=rtpmap:104 H264/90000
a=fmtp:104 max-fs=6336;max-mbps=190080;profile-level-id=42801e
a=rtcp:29241 IN IP4 172.27.14.53
```

4. Quando o *caller* finalmente recebe o 200 OK, ele transmite o ACK (Figura 2.10). A partir daí, ambos os UAs estão numa chamada estabelecida e prontos para enviarem e receberem áudio e vídeo.

Figura 2.10: SIP ACK.

```
ACK sip:UAcallee@172.27.14.53 SIP/2.0

Via: SIP/2.0/UDP 172.27.14.4:5060;branch=8dJXAX9MDw
Max-Forwards: 70
To: <sip:UAcallee@172.27.14.53>
From: <sip:UAcaller@172.27.14.4>;tag=ds17aa9bd4
Call-ID: 11022705439144@172.27.14.4
CSeq: 1 ACK
Content-Length: 0
```

5. Quando uma das partes desliga a chamada, é enviado um BYE (Figura 2.11). No caso da Figura 2.7, quem o faz é o *callee*:

Figura 2.11: SIP BYE.

```
BYE sip:UAcaller@172.27.14.4 SIP/2.0

Via: SIP/2.0/UDP 172.27.14.53:5060;branch=8dJXAX9MDv
To: <sip:UAcaller@172.27.14.4>
From: <sip:UAcallee@172.27.14.53>;tag=ds17aa9bd4
Call-ID: 11022705439144@172.27.14.4
CSeq: 5 BYE
Content-Length: 0
```

6. No momento em que o *caller* recebe o BYE, é enviado o 200 OK (Figura 2.12) para o *callee* e, desse modo, a chamada é encerrada.

Figura 2.12: SIP 200 OK.

```
SIP/2.0 200 OK

Via: SIP/2.0/UDP 172.27.14.4:5060;branch=8dJXAX9MDv
Max-Forwards: 70
To: <sip:UAcallee@172.27.14.53>
From: <sip:UAcaller@172.27.14.4>;tag=ds17aa9bd4
Call-ID: 11022705439144@172.27.14.4
CSeq: 5 BYE
Content-Length: 0
```

*2.1.2.1 **Mais informações sobre SIP***

Para mais informações sobre o protocolo SIP, consulte (IETF, 2002) e (FIRESTONE; RAMALINGAM; FRY, 2007).

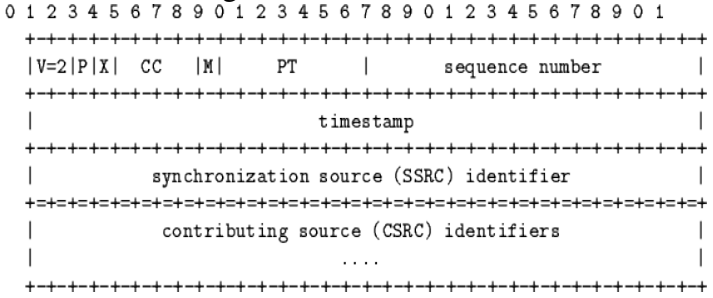
2.2 Protocolos de Transporte de Mídia em Tempo Real

Assim que os canais de mídia são estabelecidos através do Plano de Controle (durante as negociações H.323 ou SIP), é função do Plano de Mídia de uma conferência administrá-las, controlando como áudio e o vídeo serão transmitidos/recebidos e como serão tocados em um *endpoint*. Padrões específicos para o transporte de mídia em tempo real são necessários, visto que essa tarefa necessita de certos controles que protocolos de transporte comuns, como UDP e TCP, não oferecem. Neste trabalho, veremos dois desses padrões. O primeiro, RTP, definido no RFC 3550, é o protocolo de transporte em tempo real mais utilizado em aplicações multimídias atualmente. O segundo, RTMP, é o protocolo de transporte em tempo real da Adobe, usado em aplicações multimídias que utilizam a tecnologia Flash.

2.2.1 Padrão RTP

RTP - *Real-Time Protocol* - é um protocolo de nível de aplicação e, no contexto de videoconferência, é o principal padrão para a transmissão de áudio e vídeo em tempo real. O protocolo possui propriedades que auxiliam na fluidez da streaming, na sincronização e nos ajustes de qualidade e de taxa de transmissão de mídias. Um pacote RTP é composto pelo cabeçalho mais o pedaço da mídia a ser transmitido. O envio de pacotes RTP é através de um *socket* UDP, portanto todo pacote RTP é encapsulado em um segmento UDP. O cabeçalho é ilustrado na Figura 2.13 e tem a seguinte estrutura:

Figura 2.13: Pacote RTP.



Fonte: (CROWCROFT; HANDLEY; WAKEMAN, 2009).

Destacam-se os seguintes campos no cabeçalho RTP:

- *M*: é um bit de marcação. Muito usado para informar que o pedaço de áudio sendo transmitido é o começo de uma fala ou que é o início de um *frame*, caso o pacote RTP esteja transportando vídeo;

- *Payload Type*: indica qual algoritmo de codificação de mídia está sendo usado. Existem tanto algoritmos que possuem valores fixos de *Payload Type* (por exemplo, para G.722 - um tipo de codificação de áudio - o *Payload Type* é sempre 9), quanto os que possuem valores dinâmicos (que é o caso do H.264, citado nas seções anteriores), que tipicamente variam de 97 a 127. O valor do PT pode ser alterado no meio de uma transmissão de mídia, e dessa forma o receptor saberá que o algoritmo de codificação mudou. A seguir, mostramos alguns codecs e seus RTP *Payload Types* (Tabela 2.3):

Tabela 2.3: RTP Payload Types.

Codec	**Tipo**	**Payload Type**
PCMU	Áudio	0
PCMA	Áudio	8
G.722	Áudio	9
Speex	Áudio	Dinâmico
OPUS	Áudio	Dinâmico
H.261	Vídeo	31
H.263	Vídeo	34
H.264	Vídeo	Dinâmico
VP8	Vídeo	Dinâmico

- *Sequence Number*: é um contador de pacotes. A cada novo pacote RTP gerado, o *Sequence Number* é incrementado em 1. Assim, o receptor dos pacotes consegue identificar quais pedaços de mídia foram perdidos ou estão atrasados e restaurar a ordem correta desses.

- *Timestamp*: Número que define o instante de tempo em que o pacote RTP foi construído. O *Timestamp* dita o ritmo de utilização da mídia (de quanto em quanto tempo um pedaço de mídia deve ser tocado em um player, por exemplo) e ele existe pois o ritmo de recebimento dos pacotes (de quanto em quanto tempo um pedaço de mídia chega no receptor) dificilmente coincide com o de utilização.

- *Synchronization Source Identifier*: O *SSRC* é o identificador único da stream de mídia. Com isso, o receptor sabe a qual *stream* o pacote recebido pertence.

- *Contributing Source Identifiers*: O *CSRC* é a lista de identificadores que contribuíram na construção do conteúdo do pedaço de mídia que o pacote RTP carrega. Com esse campo RTP, é possível adicionar um mecanismo de mixagem em uma sessão, que é uma das tarefas do Plano de Mídia. Por exemplo, numa conferência de múltiplos usuários, um *mixer* receberá os pacotes de áudio dos vários participantes da sessão, e mixará cada *stream* de forma que cada um dos participantes receberá de volta pacotes com o áudio composto dos outros usuários (nessa composição, só não haverá o áudio do próprio participante, para ele não ouvir a si mesmo junto dos

outros). Quantos identificadores existem na lista *CSRC* é definido no campo *CC* (*CSRC count*), localizado no começo do cabeçalho RTP.

2.2.1.1 ***RTCP***

O RTCP - *RTP Control Protocol* - é o protocolo que atua junto ao RTP. Tem como principal função fornecer aos usuários *feedback* sobre atrasos de pacotes e suas variações, uso e congestionamento de banda, e outras propriedades da rede. Também pode atuar em sincronização de mídia, nomeações de *streams* e identificação de participantes. RTP e RTCP atuam em portas diferentes, a fim da identificação do que é um dado de mídia (no contexto de videoconferência), e o que é um dado de controle.

Tais *feedbacks* são úteis para adaptações nas transmissões. Se pacotes RTCP indicam que a rede não está enfrentando nenhum problema, o transmissor da mídia tem a oportunidade de, por exemplo, trocar para algum codec que ofereça mais qualidade mas que ocupe mais banda, ou simplesmente, utilizando o mesmo codec, elevar a taxa de transmissão a fim de aumentar a qualidade da mídia. Da mesma forma, caso o transmissor receba informações RTCP que apontam problemas na rede, é possível diminuir o uso de banda, alterando codecs e taxas de transmissão (o que provavelmente irá diminuir a qualidade da mídia). O uso alternado e *on-the-fly* de codecs durante a sessão é possível modificando o *Payload Type* dos pacotes RTP (antes do momento efetivo da troca, há nova negociação de mídia SIP ou H.323).

Feedbacks RTCP também são úteis para geração de relatórios que apontam dados importantes como porcentagem de pacotes perdidos, flutuação da qualidade da mídia durante o andamento da sessão, entre outros.

Uma questão envolvendo RTCP é o uso de banda pelo próprio. Pacotes RTCP são enviados para todos os participantes de uma sessão, de forma que o aumento do grupo de usuários pode também levar a um rápido crescimento do uso de banda pelo protocolo RTCP. Diminuir a taxa de envio de *feedbacks* de forma que o envio de mensagens RTCP não ocupe mais que 5 % da banda é a norma para evitar tal situação.

Para mais informações sobre RTP e RTCP, consulte (SCHULZRINNE et al., 2003).

2.2.2 Padrão RTMP

Adobe's Real Time Messaging Protocol (RTMP) é o protocolo de transporte usado para *streaming* de áudio, vídeo e outros tipos de dados da plataforma Flash. Inicialmente um padrão proprietário da Macromedia, a Adobe - assim que comprou a empresa - tornou a especificação do padrão pública, visando a interoperação de outros sistemas com seus produtos. Mensagens RTMP são utilizadas somente no nível de aplicação e, diferentemente de RTP - que roda sobre UDP - pacotes RTMP são enviados através de uma conexão TCP, normalmente usando a porta 1935. Uma mensagem RTMP é formada pelo cabeçalho e pelo pedaço de mídia a ser transmitido (no contexto de videoconferência). O cabeçalho de uma mensagem RTMP deve conter:

- *Timestamp*;
- *Length*: tamanho do *payload* (que contém os dados de mídia);
- *Type Id*: tipo da mensagem que o pacote RTMP carrega. Define se o conteúdo é áudio (*Type Id* = 8), vídeo (*Type Id* = 9), ou controle (há um conjunto de *Type Ids* específicos para o controle de *streaming*).

- *Message Stream ID*: número arbitrário que define o identificador único da *stream*.

Após a montagem da mensagem, ele sofre o processo de *chunking*, onde a mensagem é fragmentada em pedaços menores, os *chunks*. *Chunks* possuem cabeçalho e *payload* variáveis, diminuindo, assim, o *overhead* gerado por pacotes de tamanho fixo (pequenos dados a serem transmitidos em pacotes de tamanho fixo às vezes devem preencher o payload ou cabeçalhos com *dummy bits* apenas para respeitar o tamanho fixo ; assim, parte do pacote a ser transmitido é inútil, e, desse modo, se usa mais banda que o necessário). O receptor, por sua vez, agrupa os *chunks* de uma mesma mensagem através do *Chunk Stream Id*, campo do cabeçalho do *chunk*, montando, assim, a mensagem original.

Para mais informações sobre o protocolo RTMP, consulte (PARMAR; THORNBURGH, 2012).

2.3 Codificação de Mídia

Para diminuir o uso de banda da transmissão de mídia, áudio e vídeo sofrem um processo de compressão antes de serem encapsulados em pacotes RTP ou RTMP. As características de um algoritmo de compressão determinam uma maior ou menor redução de dados, bem como ditam o quanto a mídia perderá qualidade no processo de compressão. Codificação/decodificação - *codec* - (comprimir e descomprimir), pode ser feito tanto em software quanto em hardware, de forma que sistemas de videoconferência de sala tendem a levar vantagem no desempenho dessa tarefa, devido ao seu hardware dedicado. Codecs são *lossy* quando a compressão provoca perda de dados, ou, caso contrário, *lossless*.

Um parâmetro de codificação importante - tanto para áudio quanto para vídeo - é a taxa de bits ou *bitrate*, que define quantos bits são processados para reproduzir uma unidade de tempo da mídia. Por exemplo, se uma compressão de áudio gera uma taxa de 1kbps, a cada 1 segundo de áudio há 1000 bits. Assim, a taxa de bits de uma codificação impacta o tamanho total da mídia (e, consequentemente, o uso da banda para transmiti-la), sua qualidade (em geral, quanto mais bits por unidade de tempo, mais fiel a codificação é à mídia original - embora haja codecs de baixo *bitrate* e boa qualidade), e seu processamento (quanto menor a taxa de bits, maior é a demanda da compressão para reduzir a mídia). Uma codificação pode gerar uma taxa de bits constante (CBR - *Constant bitrate*) ou variável (VBR - *Variable bitrate*).

Para áudio, podem ser destacados outros parâmetros e características, como:

- Taxa de amostragem (ou *sample rate*): o codec pode alterar o *sample rate* (número de amostras por segundo usado na captura do áudio original), determinando um novo espectro de frequências do áudio e impactando na qualidade. No contexto de videoconferência, codecs usam *sample rates* que variam de 8kHz a 48kHz;

- Supressão de silêncio - *silence suppression* ou *Voice Activity Detection* (*VAD*): o codec pode detectar silêncios no áudio e descartar os dados, reduzindo o tamanho da mídia;

- Cancelamento de eco: o codec pode detectar ecos (geralmente causados por microfones que acabam capturando a saída de alto-falantes e remandando o áudio que havia chegado para outros participantes) e os cancelar;

Já para vídeo, aponta-se:

- Quadros por segundos (FPS - *frames per second*): número de imagens que serão exibidas no intervalo de 1 segundo. Quanto maior o FPS, maior a fluidez do vídeo e maior uso de banda em transmissão (é uma relação linear: dobrando o FPS, dobra-se a ocupação da banda);

- Resolução: largura e altura (em pixels) dos quadros que compõe o vídeo. Resoluções conhecidas são SD (720x480), HD (1280x720) e Full-HD (1920x1080). Possui relação quadrática com o uso de banda, de forma que dobrar a resolução (aumentar a largura e comprimento cada um em duas vezes) quadruplica a ocupação da banda;

- Perfil ou *profile*: define um conjunto de características que limita a complexidade da codificação, e, consequentemente, reduz a demanda da decodificação. Por exemplo, alguns perfis podem proibir o uso de *B-Frame* (quadro que tem alta compressão, pois usa informações de quadros anteriores e próximos), pois estes demandam mais processamento e uso de memória;

- *Level*: define limitações numéricas de parâmetros, como resolução, *fps*, *bitrate*, etc.

A seguir, veremos resumidamente alguns padrões populares e/ou atuais usados para a codificação de áudio e vídeo em sistemas de videoconferência:

2.3.1 Padrões para Codificação de Áudio

2.3.1.1 ***Padrão G.7xx***

Codecs da família G.7xx (ou G Series) são padrões ITU-T.

O origem do G Series se deu em 1972, com o lançamento do padrão G.711, formalmente conhecido como *Pulse Code Modulation* (PCM) *of Voice Frequencies*. Foi primeiramente usado em telefonia, onde existiam 2 versões: o G.711 μ-law (ou PCMU), usado em redes de telefonia dos Estados Unidos e Japão, e o G.711 A-law (ou PCMA) usado na Europa e internacionalmente (TANENBAUM; WETHERALL, 2011). Ainda houve melhorias, como o G.711.0, que gerava compressão sem perdas (*lossless*) e como o G.711.1, que incrementava a qualidade do áudio (bem como o uso de banda). O suporte de G.711 é obrigatório em sistemas H.323.

A Tabela 2.4 mostra os *bitrates* e a qualidade de alguns codecs G.7xx (OGUNFUNMI; NARASIMHA, 2010). A qualidade foi medida usando MOS, popular tipo de medida subjetiva de qualidade de áudio, classificando-a de 1 a 5 (MOS não será abordado nesse trabalho). Nessa tabela se observa que, no geral, a evolução da G Series apresentou uma taxa de compressão cada vez maior (diminuindo o *bitrate* a cada nova versão, exceto nas versões G.727 e G.728), enquanto manteve o nível de qualidade (exceto pelo G.723.1, em que, comparado à versão anterior deste, reduziu-se bruscamente o *bitrate*).

Mais informações sobre G.7xx é possível acessando *ITU-T G Series: Transmission systems and media, digital systems and networks*[1], página web da ITU-T dedicada a G Series, contendo todas as especificações.

[1] `http://www.itu.int/net/itu-t/sigdb/speaudio/Gseries.htm`

Tabela 2.4: Comparação G Series.

Codec	Bitrate (kbps)	Qualidade
G.711	64	4.3
G.721/G.726	16,24,32,40	4.1
G.722	32	4.1
G.722.1	32	4.1
G.722.2	32	4.1
G.723.1	5.3 até 6.3	3.7 até 3.9
G.727	8	4.3
G.728	16	4.0
G.729	8	4.3
G.729A	8	4.3

*2.3.1.2 **Padrão Speex e Padrão OPUS***

Speex e OPUS são dois padrões da fundação Xiph.Org[2], organização sem fins lucrativos que desenvolve padrões de software open source.

Speex tornou-se um padrão IETF no RFC 5574[3] e se baseia no algoritmo de compressão CELP (*Code-Excited Linear Prediction*). Algumas características do Speex são (XIPH.ORG, 2003):

- *Lossy*;
- *Sample rate* de 8, 16 ou 32 kHz;
- *Bitrate* variando de 2.15 a 44.2 kbps;
- VBR;
- Cancelamento de eco, supressão de ruído e de silêncio.

Em 2012, surge o OPUS, como uma espécie de 'sucessor' do Speex (o próprio speex.org[4] declara na sua home page: "The Speex codec has been obsoleted by Opus. It will continue to be available, but since Opus is better than Speex in all aspects, users are encouraged to switch."), incorporando características do CELT (outro codec de áudio da Xiph.Org) e SILK (codec de áudio utilizado no Skype).

OPUS tornou-se um padrão IETF no RFC 6716[5] e é o principal codec de áudio da tecnologia WebRTC (Seção 2.4.2), conseguindo baixos *bitrates* e alta qualidade. O desenvolvimento do OPUS recebeu colaborações de programadores associados ao Xiph.Org e ao Skype.

OPUS apresenta algumas features como (XIPH.ORG, 2012):

- *Lossy*;
- *Sample rate* de 8 a 48 kHz;

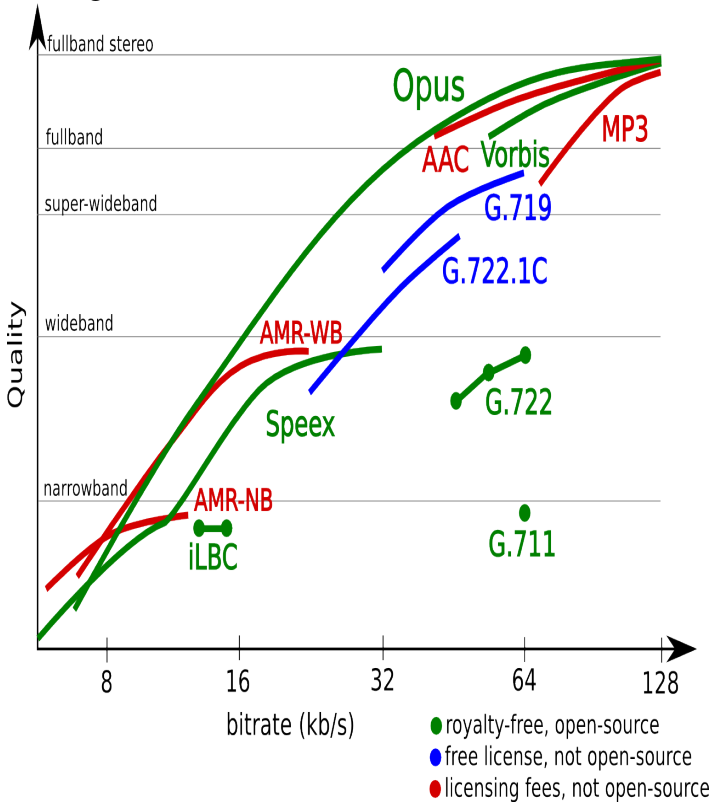
[2] https://www.xiph.org/about/
[3] https://tools.ietf.org/html/rfc5574
[4] https://www.speex.org
[5] https://tools.ietf.org/html/rfc6716

- *Bitrate* variando de 6 a 510 kbps;

- Suporta tanto VBR quanto CBR;

O opus-codec.org[6] oferece uma página de comparações entre OPUS e outros codecs de áudio. O gráfico abaixo (Figura 2.14), mostra a relação de *bitrate* e qualidade de áudio em diversos codecs:

Figura 2.14: OPUS vs outros codecs de áudio.



Fonte: (XIPH.ORG, 2012).

2.3.1.3 *Padrão iLBC e Padrão iSAC*

Internet Low Bitrate Codec (iLBC) e *internet Speech Audio Codec* (iSAC) são codecs de áudio desenvolvidas pela Global IP Solutions, comprada pela Google em 2011, sendo utilizados no Google Hangout e posteriormente incorporados ao WebRTC.

Segundo o overview da arquitetura do WebRTC[7], iSAC apresenta *sample rate* de 16 ou 32 kHz, com um *bitrate* variável de 12 a 52 kbps.

Já iLBC tornou-se padrão IETF, e é definido nos RFCs 3951 e 3952. Algumas características são seu *sample rate* de 8 kHz e seu *bitrate* de 13.33 kbps ou 15.2kbps. Segundo (WEBRTC, 2011), a qualidade básica do áudio usando iLBC é maior que a oferecida pelo ITU-T G.729A.

[6] `http://opus-codec.org/comparison/`
[7] `https://webrtc.org/architecture/`

2.3.2 Padrões para Codificação de Vídeo

2.3.2.1 ***Padrão H.26x***

Codecs H.26x são padrões ITU-T para video. No contexto de videoconferência, destacam-se:

- **H.261**: Usado para interoperação com *endpoints* antigos (*legacy*). Vídeos H.261 podem ser QCIF (resolução de 176x144) a 30 *fps* ou CIF (352x288) a 15 *fps* (IETF, 2006).

- **H.263**: Existem 3 versões para H.263: *Base* H.263 (ou, simplesmente, H.263), H.263+ ou H.263-1998 (que também engloba as características do H.263) e H.263++ ou H.263-2000 (também englobando as características do H.263-1998).

  H.263 define as seguintes resoluções: sub-QCIF (128x96), QCIF, CIF, CIF4 (704x576), CIF16 (1056x864) e CUSTOM (personalizável - largura e altura devem ser múltiplos de 4).

  H.263 e H.263-1998, que não suportam *profiles* e *levels*, utilizam, por default, QCIF a 30 *fps* (podendo utilizar outros valores através de negociações SIP ou H.323).

  H.263-2000 apresenta 9 perfis de vídeo e múltiplos *levels*, que definem quais os *bitrates*, *fps* e resoluções permitidas. Para videoconferência, o profile recomendado é o *baseline* (*profile* 0), que é o perfil 'mínimo' (sem as *features* adicionais e opções de operação de outros perfis).

- **H.264**: Sucessor do H.263. Foi desenvolvido em conjunto com a ISO/IEC, sendo também conhecido como AVC (*Advanced Video Codec*). Gera de 30 a 50% menos *bitrate* que H.263, embora aumente o uso de CPU - pode demandar 3 ou 4 vezes mais processamento que outros codecs de vídeo (FIRESTONE; RAMALINGAM; FRY, 2007), dependendo do *profile* usado na codificação H.264.

  O perfil H.264 para videoconferência é o *baseline*. *Levels* válidos são: 1, 1.1, 1.2, 1.3, 2, 2.1, 2.2, 3, 3.1, 3.2, 4, 4.1, 4.2, 5, e 5.1. Por exemplo, o level 3.0 define 4CIF a 25 fps. É possível consultar a lista completa de *levels* e suas definições em (ITU, 2016).

2.3.2.2 ***Padrão VPx***

O padrão de codificação VPx foi desenvolvida pela On2 Technologies, originado da série de codecs *True Motion*. O primeiro padrão VPx foi o *True Motion* VP3, sucessor do *TrueMotion* RT 2.0.

Em 2010, a On2 Technologies foi adquirida pela Google[8] e, assim, o padrão VP8, que fora publicado em 2008, tornou-se público através do WebM Project[9], sendo posteriormente incorporado ao pacote de mídia do WebRTC[10].

O codec VP8 define 4 *profiles* e 3 *levels* (IETF, 2011) que, diferentemente da definição convencional - explicado e exemplificado anteriormente - apenas regulam a complexidade do processo de decodificação/descompressão dos dados de vídeo.

[8] `http://www.on2.com/`

[9] `http://www.webmproject.org/about/`

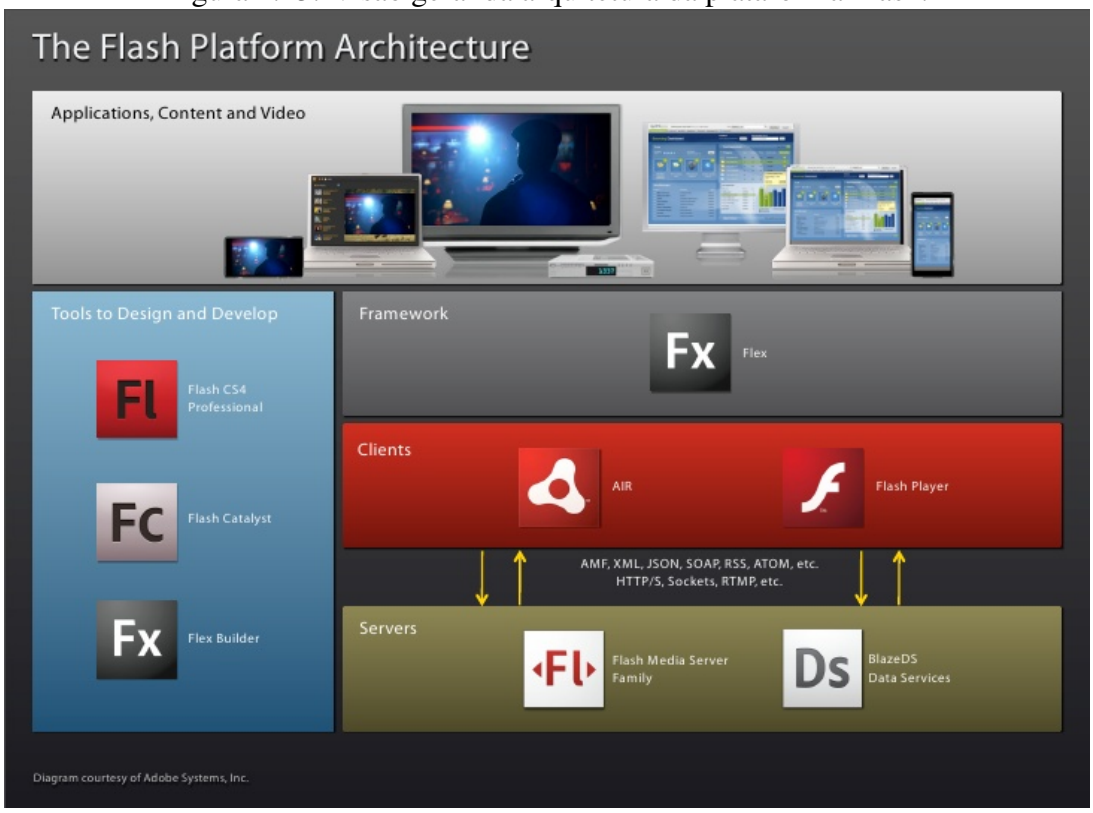
[10] `https://webrtc.org/faq/#what-is-the-vp8-video-codec`

2.4 Tecnologias para Webconferência

2.4.1 Plataforma Flash

Adobe Flash (anteriormente Macromedia Flash) é uma plataforma de desenvolvimento de software para desktop, mobile e navegadores web. Na Figura 2.15, temos a visão geral da sua arquitetura.

Figura 2.15: Visão geral da arquitetura da plataforma Flash.



Fonte: (ADOBE, 2016).

O desenvolvimento de aplicações Flash se dá através da linguagem de programação ActionScript[11] e, para a confecção de interfaces de usuário, da linguagem de marcação MXML[12], ambas integradas no framework Flex[13]. Para administrar os projetos de código que utilizam tal framework, há a IDE Flex Builder[14], renomeada posteriormente para *Flash Builder* (exclusiva para Windows). O software Adobe Air[15] permite que o mesmo código seja executado em diferentes ambientes (Windows, Linux, Android, Mac OS, entre outros).

É através do software Flash Player que navegadores web podem, por meio da instalação do plugin, reproduzir conteúdo multimídia. Devido sua facilidade de uso, logo tornou-se um padrão popular para *streaming* de áudio e vídeo na web. Entre os famosos provedores de conteúdo que utilizam ou já utilizaram o Flash Player, destacam-se Spotify, Yahoo, BBC Online, Youtube, entre outros.

[11] http://www.adobe.com/devnet/actionscript.html
[12] http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf5f39f-7fff.html
[13] http://www.adobe.com/br/products/flex.html
[14] http://www.adobe.com/br/products/flash-builder.html
[15] http://www.adobe.com/br/products/air.html

Sistemas de webconferência desenvolvidas sobre a plataforma Flash devem usar os padrões suportados pelo Flash Player que, no contexto de videoconferência, define o seguinte escopo:

- Transporte de mídia: RTMP;
- Codecs de áudio: Speex e G.711 (PCMA e PCMU);
- Codecs de vídeo: H.263, H.264 e VP6.

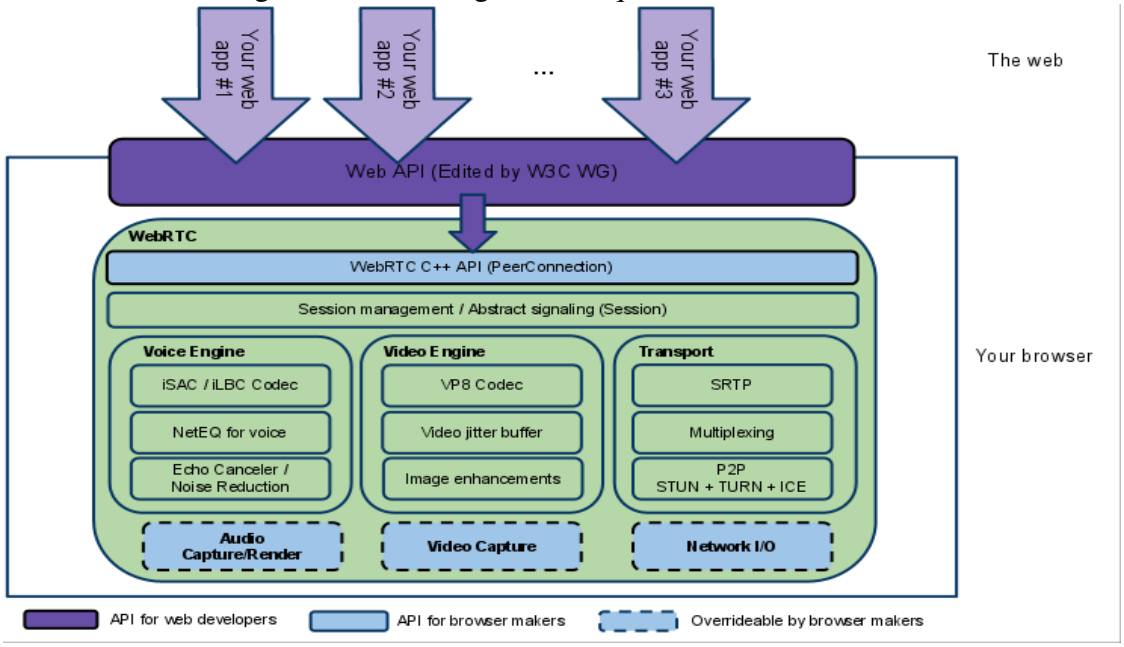
Depende-se de um servidor de mídia Flash para a distribuição de áudio e vídeo entre múltiplos clientes Flash. A conexão destes com o servidor se dá através de um *NetConnection*, definida na API do ActionScript. O servidor de mídia Flash também pode conectar usuários externos (não-Flash) que utilizam SIP, já que o Flash Player por si só não define Plano de Controle. Servidores de mídia Flash conhecidos são o Adobe Media Server[16] (ilustrado na Figura 2.15 como *Flash Media Server*) e o Red5 (que veremos no próximo capítulo).

2.4.2 WebRTC

WebRTC é um framework aberto que provê capacidades de comunicação em tempo real para navegadores web, implementando componentes que dão suporte a chat, áudio e vídeo, sem a necessidade de instalação de plugins. Desenvolvedores que desejam utilizar WebRTC para construir um sistema de webconferência podem acessar tais componentes pela Web API, que utiliza JavaScript. À época deste trabalho, WebRTC é suportado pelos navegadores web Chrome, Firefox e Opera, e está em processo de se tornar um padrão IETF.

A arquitetura geral do WebRTC é apresentada Figura 2.16.

Figura 2.16: Visão geral da arquitetura do WebRTC.



Fonte: (WEBRTC, 2016).

[16] http://www.adobe.com/br/products/adobe-media-server-family.html

A camada em verde é a camada implementada pelos navegadores web. Eles devem prover a camada RTP e a implementação dos codecs de mídia. Navegadores web devem, no mínimo, implementar VP8, OPUS, iSAC e iLBC. Chrome e Firefox ainda oferecem suporte a H.264.

A conexão entre clientes WebRTC se dá através de um *WebSocket*, tecnologia que utiliza TCP para conectar navegadores web. Assim como o Flash Player, WebRTC não define Plano de Controle. A filosofia do WebRTC é deixar a escolha do uso do protocolo de sinalização para a aplicação web, enquanto o navegador web fica responsável pelo Plano de Mídia. Entretanto, o próprio WebRTC faz recomendações de bibliotecas web que implementam SIP, como o JsSIP[17] e o SIP.js[18], ambas implementadas em JavaScript. Assim, um cliente WebRTC tem a capacidade de fazer ligações para *endpoints* ou servidores SIP.

Informações adicionais sobre WebRTC é possível acessando o webrtc.org[19].

[17] `http://www.jssip.net/`
[18] `http://sipjs.com/`
[19] `https://webrtc.org/`

3 SOLUÇÕES *OPEN SOURCE* PARA INTEROPERABILIDADE

Como vimos no capítulo 2, existem diversos padrões para videoconferência. Padrões que desempenham a mesma função (sinalizar, transportar ou codificar) apresentam muitas propriedades em comum. Por exemplo, ambos SIP e H.323 definem mensagens que estabelecem e encerram chamadas, negociam mídias, etc. Protocolos de transporte em tempo real apresentam campos de *timestamp*, mecanismos de identificação de dados (se mídia ou controle), entre outros. Portanto, não é surpresa que haja soluções que se propõem a fazer o meio de campo entre esses padrões, efetuando a tradução de um para outro. A seguir, então, veremos algumas soluções open source e quais problemas de interoperabilidade elas são capazes de resolver. Exploraremos com detalhes 3 soluções: Freeswitch, FFMPEG e Red5. Ao final do capítulo, ainda, são citadas mais duas soluções, visando trabalhos futuros: Asterisk e Kurento.

3.1 Freeswitch

Freeswitch[1] é um *softswitch*. Um switch, no sentido de telefonia, é um dispositivo (hardware dedicado) de rede de telecomunicação que conecta e faz o roteamento de ligações. Um softswitch é um switch em software.

As configurações do Freeswitch são definidas em arquivos XML. Dentre as inúmeras características do Freeswitch (FS), destacam-se:

- Configuração de *dialplans*: Através de *dialplans*, definimos as regras de ligações e de roteamentos. Por exemplo, podemos definir que as ligações que contêm 5 dígitos serão encaminhadas para uma sala de conferência, como define o dialplan da Figura 3.1;

Figura 3.1: Exemplo de um *dialplan* em XML.

```
1  <include>
2      <extension name="example">
3        <condition field="destination_number" expression="^(\d{5})$">
4          <action application="set"/>
5          <action application="answer"/>
6          <action application="conference" data="$1@cdquality"/>
7        </condition>
8      </extension>
9  </include>
10
```

[1] https://freeswitch.org/

- Registro de usuários: Um exemplo de registro encontra-se na Figura 3.2. Com esse tipo de recurso é possível, por exemplo, definir *dialplans* onde o roteamento da ligação só ocorrerá se o usuário que está ligando ou sendo chamado for registrado;

Figura 3.2: Registo de usuário básico.

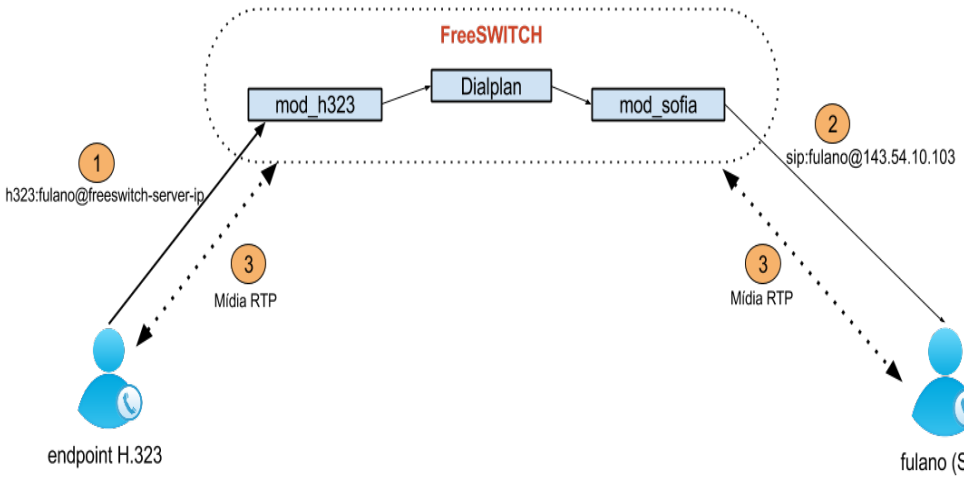
```
1  <domain name="inf.ufrgs.br">
2    <user id="alexandre">
3      <params>
4        <param name="password" value="@inf"/>
5      </params>
6    </user>
7  </domain>
```

- Capacidade de conectar múltiplos participantes: através do *mod_conference*, módulo de conferência do FS (que será detalhado mais adiante), é possível que múltiplos usuários se conectem em uma sessão via SIP ou H.323. *mod_conference* irá administrar essa sessão, gerenciando os participantes e sendo um servidor de mídia, portanto, distribuindo áudio e vídeo entre o grupo de *endpoints*.

O Freeswitch provê recursos para resolver os seguintes problemas de interoperabilidade:

- **Conectar endpoints SIP e H.323**: Freeswitch tem 2 módulos específicos para lidar com endpoints SIP e H.323. *mod_sofia* é a implementação da camada SIP e *mod_h323*, a da camada H.323. A Figura 3.3 ilustra um exemplo de como acontece a ponte H.323/SIP.

Figura 3.3: Ponte H323-SIP do Freeswitch.



Começamos com um endpoint H.323 ligando para o Freeswitch, pedindo para se conectar ao usuário *fulano* (item 1 na figura). Quem lida com a ligação (e com todas as mensagens H.323) é o *mod_h323*. Após, este repassa o destino (*fulano*) para o módulo responsável pelos *dialplans*. O Freeswitch irá checar suas configurações e deverá encontrar um *dialplan* com algum formato desse tipo (Figura 3.4):

Figura 3.4: Exemplo de *dialplan* de redirecionamento SIP.

```
<include>
    <extension name="bridge-example">
      <condition field="destination_number" value="fulano">
        <action application="bridge" data="sofia/external/
        fulano@143.54.10.103"/>
      </condition>
    </extension>
</include>
```

Nele, o Freeswitch confirma que, se o destino for *fulano*, ele deverá fazer uma ligação SIP para o IP 143.54.10.103 (item 2), realizada a partir do *mod_sofia*. A partir do momento em que *fulano* aceita a ligação e ambos os *endpoints* estão conectados, o Freeswitch recebe e repassa os dados de mídia para as partes envolvidas (3) via RTP, podendo haver transcodificação de codecs (o que veremos no próximo item).

É graças a esse mecanismo que o *mod_conference* pode conectar endpoints SIP e H.323 na mesma sessão. Ao receber uma ligação SIP ou H.323, *mod_sofia* ou *mod_h323* redirecionam a chamada para o *mod_conference* através de um dialplan como o da Figura 3.1.

- **Transcodificar codecs de áudio**: o Freeswitch é capaz de conectar usuários que não implementam os mesmos codecs de áudio. Se, ainda considerando a Figura 3.3, o endpoint H.323, ao fazer a ligação, afirmasse via protocolo H.245 que só trabalha com áudio G.711, e o *fulano*, ao aceitá-la, afirmasse via SDP que apenas implementa Speex, o Freeswitch conectaria os usuários realizando a transcodificação dos áudios. No exemplo, FS receberia áudio G.711, transcodificaria a mídia para Speex, e a repassaria para *fulano*; da mesma forma, receberia Speex, transcodificaria tal codec para G.711, e repassaria o resultado para o endpoint H.323. A lista de codecs de áudio que o FS é capaz de transcodificar é descrita no arquivo *vars.xml*, que é o local onde há a declaração das variáveis usadas globalmente nas diversas configurações possíveis (dialplans, registro de usuários, entre outros). A lista completa está na Tabela 3.1. No mesmo arquivo, é listado os codecs que o Freeswitch não tem a capacidade de transcodificar (*passthrough*), ou seja, o Freeswitch só tem a capacidade de repassar para os *endpoints* as mídias que usam tais codecs. Basicamente, os codecs *passthrough* do Freeswitch são os codecs de vídeo e também alguns poucos de áudio. A lista está na Tabela 3.2.

Tabela 3.1: Lista de codecs que o Freeswitch é capaz de transcodificar.

Lista de codecs transcodificáveis em *vars.xml*
opus@48000h@10i - Opus 48khz using 10 ms ptime (mono and stereo)
opus@48000h@20i - Opus 48khz using 20 ms ptime (mono and stereo)
opus@48000h@40i - Opus 48khz using 40 ms ptime
opus@8000h@10i - Opus 8khz using 10 ms ptime (mono and stereo)
opus@8000h@20i - Opus 8khz using 20 ms ptime (mono and stereo)
opus@8000h@40i - Opus 8khz using 40 ms ptime
opus@8000h@60i - Opus 8khz using 60 ms ptime
opus@8000h@80i - Opus 8khz using 80 ms ptime
opus@8000h@100i - Opus 8khz using 100 ms ptime
opus@8000h@120i - Opus 8khz using 120 ms ptime
iLBC@30i - iLBC using mode=30 which will win in all cases.
DVI4@8000h@20i - IMA ADPCM 8kHz using 20ms ptime. (multiples of 10)
DVI4@16000h@40i - IMA ADPCM 16kHz using 40ms ptime. (multiples of 10)
speex@8000h@20i - Speex 8kHz using 20ms ptime.
speex@16000h@20i - Speex 16kHz using 20ms ptime.
speex@32000h@20i - Speex 32kHz using 20ms ptime.
BV16 - BroadVoice 16kb/s narrowband, 8kHz
BV32 - BroadVoice 32kb/s wideband, 16kHz
G7221@16000h - G722.1 16kHz (aka Siren 7)
G7221@32000h - G722.1C 32kHz (aka Siren 14)
CELT@32000h - CELT 32kHz, only 10ms supported
CELT@48000h - CELT 48kHz, only 10ms supported
GSM@40i - GSM 8kHz using 40ms ptime. (GSM is done in multiples of 20, Default is 20ms)
G722 - G722 16kHz using default 20ms ptime. (multiples of 10)
PCMU - G711 8kHz ulaw using default 20ms ptime. (multiples of 10)
PCMA - G711 8kHz alaw using default 20ms ptime. (multiples of 10)
G726-16 - G726 16kbit adpcm using default 20ms ptime. (multiples of 10)
G726-24 - G726 24kbit adpcm using default 20ms ptime. (multiples of 10)
G726-32 - G726 32kbit adpcm using default 20ms ptime. (multiples of 10)
G726-40 - G726 40kbit adpcm using default 20ms ptime. (multiples of 10)
AAL2-G726-16 - Same as G726-16 but using AAL2 packing. (multiples of 10)
AAL2-G726-24 - Same as G726-24 but using AAL2 packing. (multiples of 10)
AAL2-G726-32 - Same as G726-32 but using AAL2 packing. (multiples of 10)
AAL2-G726-40 - Same as G726-40 but using AAL2 packing. (multiples of 10)
LPC - LPC10 using 90ms ptime (only supports 90ms at this time in FreeSWITCH)
L16 - L16 isn't recommended for VoIP but you can do it. L16 can exceed the MTU rather quickly.

Tabela 3.2: Lista de codecs que o Freeswitch não é capaz de transcodificar.

Lista de codecs não transcodificáveis em *vars.xml*
G729 - G729 in passthru mode. (mod_g729)
G723 - G723.1 in passthru mode. (mod_g723_1)
AMR - AMR in passthru mode. (mod_amr))
H261 - H.261 Video
H263 - H.263 Video
H263-1998 - H.263-1998 Video
H263-2000 - H.263-2000 Video
H264 - H.264 Video
VP8 - VP8 Video

Apesar de dar suporte a todos os codecs acima (sejam *passthrough* ou não), é possível configurar quais deles serão aceitos nas negociações SIP e H.323, através da variável *global_codec_prefs* em *vars.xml* (Figura 3.5).

Figura 3.5: Exemplo da variável *global_codec_prefs* em *vars.xml*.

```
134     <X-PRE-PROCESS cmd="set" data="global_codec_prefs=OPUS,speex@16000h@20i,
        speex@8000h@20i,G7221@32000h,G7221@16000h,G722,PCMU,PCMA,GSM,H264" />
```

3.1.1 *mod_conference*

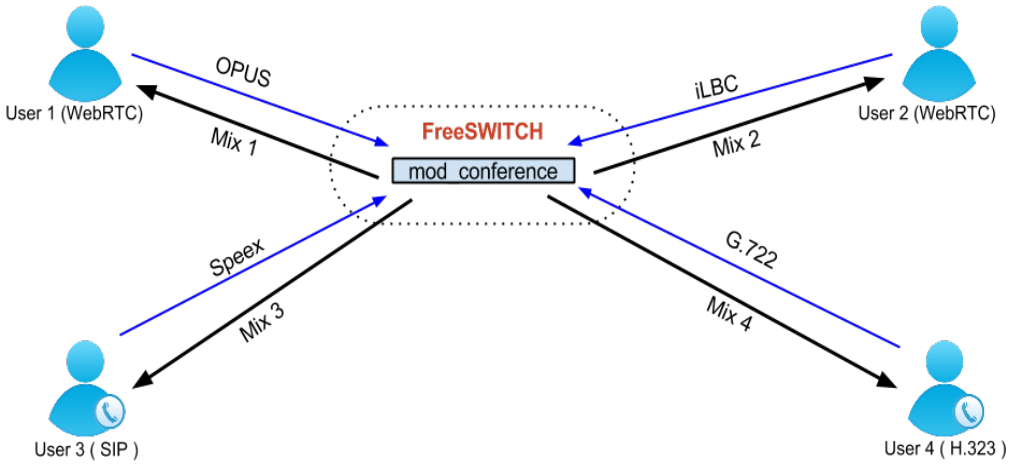
Como explicado anteriormente, *mod_conference* é o módulo de conferência do Freeswitch, capaz de conectar múltiplos usuários e de ser um servidor de mídia, distribuindo as mídias ativas entre os participantes de uma sessão.

No exemplo de dialplan da Figura 3.1, quando o FS recebe uma ligação de 5 dígitos, ocorre o redirecionamento para uma sala de conferência. Quando um primeiro usuário liga, por exemplo, *sip:12345@freeswitch-server-ip* e a chamada chega no Freeswitch, o dialplan, verificando que o número discado é de 5 dígitos, criará uma sala de conferência chamada *12345* e conectará o usuário nessa sala. Todo endpoint que ligar posteriormente *sip:12345@freeswitch-server-ip* ou *h323:12345@freeswitch-server-ip* será direcionado para a sala *12345*.

Uma importante questão é a política do servidor de mídia para a distribuição desta entre o grupo conectado. O Freeswitch apresenta duas políticas distintas para áudio e vídeo.

Para áudio, ocorre *mixagem*. O exemplo da Figura 3.6 demonstra o mecanismo dessa política.

Figura 3.6: Mixagem de áudio do Freeswitch.

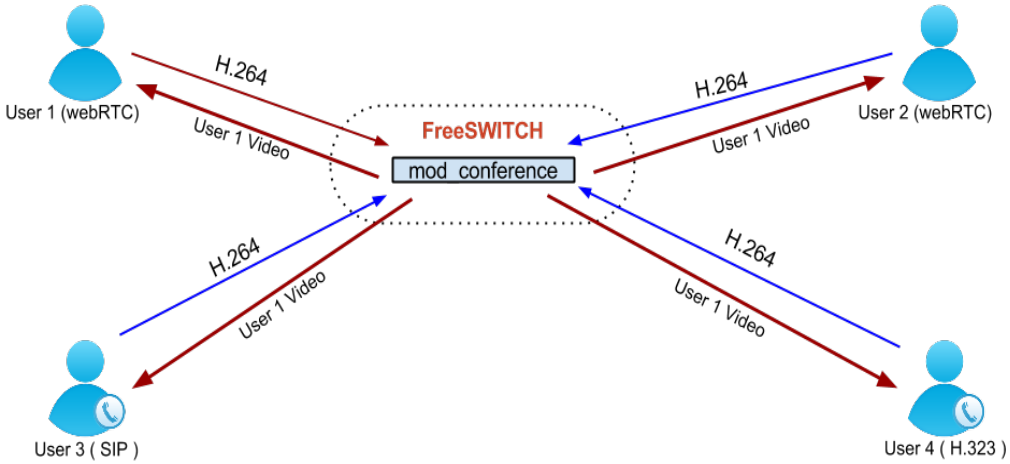


No exemplo, temos quatro *endpoints*: dois (User 1 e User 2) utilizando uma aplicação WebRTC (que permite que usuários se conectem no Freeswitch através do SIP.js, por exemplo), um *endpoint* genérico SIP - equipamento ou softfone - (User 3), e outro *endpoint* genérico H.323 (User 4). A mixagem de áudio no Freeswitch ocorre da seguinte maneira: o *mod_conference* recebe os pacotes de áudio do User 2, 3 e 4, transcodifica-os para OPUS, e executa a composição dos pacotes numa única *stream* RTP chamada Mix 1.

Após, o Mix 1 é transmitido para o User 1, que, desse modo, ouvirá todos os participantes, menos a si próprio. Assim ocorre com todos os demais participantes: User 2 recebe o Mix 2 em iLBC, composto dos áudios dos usuários 1,3,4 ; User 3 recebe o Mix 3 em Speex, composto dos áudios dos usuários 1,2,4 ; e, finalmente, User 4 recebe o Mix 4 em G.722, composto dos áudios dos usuários 1,2,3.

Para vídeo, ocorre *switching*. O vídeo que é distribuído para o grupo é o vídeo do participante que está falando no momento, o chamado *video floor*. No exemplo da Figura 3.7, podemos ver o vídeo do User 1 sendo transmitido para todos os usuários (incluindo ele próprio). Portanto, o User 1 é o usuário com áudio ativo no momento.

Figura 3.7: *Switching* de vídeo do Freeswitch.



3.1.1.1 ***Freeswitch 1.6***

À época do desenvolvimento deste trabalho, foi lançado o Freeswitch Versão 1.6, prevendo várias *features* novas para a distribuição de vídeo, como:

- Composição de vídeo: o Freeswitch recebe os vídeos de todos os usuários e constrói uma *stream* única de vídeo, em que seu conteúdo é a composição de todos os vídeos recebidos. Assim, é transmitida a mesma *stream* de vídeo para os usuários. Cada um verá, dessa maneira, a imagem dos outros participantes e também a si mesmo. Um exemplo de composição é mostrado na Figura 3.8.

- Configuração de *Layouts*: um vídeo composto pode dar ênfase a determinados vídeos através de *layouts*, como pode ser visto na Figura 3.9.

- Seleção de vídeos: Um usuário pode escolher qual vídeo, dentre os vídeos que o Freeswitch está recebendo no momento, ele quer visualizar.

Para mais informações, é possível acessar a página do Freeswitch 1.6 Video[2].

[2] `https://freeswitch.org/confluence/display/FREESWITCH/FreeSWITCH+1.6+Video`

Figura 3.8: Exemplo de composição de vídeo do Freeswitch.

Fonte: (FREESWITCH, 2016).

Figura 3.9: Quatro exemplos de *layouts* de vídeo do Freeswitch 1.6.

Fonte: (FREESWITCH, 2016).

3.2 FFMPEG

O projeto FFMPEG[3] é um *framework* multimídia, capaz de codificação, transcodificação, transmissão, análise e reprodução de áudio e vídeo. O projeto apresenta 4 ferramentas:

- *ffmpeg*[4]: ferramenta de linha de comando que tem a capacidade de converter mídias entre diferentes formatos;
- *ffserver*[5]: servidor de mídia (recebimento e distribuição de áudio e vídeo);
- *ffplay*[6]: player de mídia;

[3] `https://ffmpeg.org`
[4] `https://ffmpeg.org/ffmpeg.html`
[5] `https://ffmpeg.org/ffserver.html`
[6] `https://ffmpeg.org/ffplay.html`

- *ffprobe*[7]: ferramenta que levanta dados de uma stream específica, como codecs usados, *bitrate*, *sample rate*, resolução, *fps*, etc.

Este trabalho focará na ferramenta *ffmpeg*, visando resolver questões de interoperabilidade que o Freeswitch não prevê, que são:

- Transcodificação RTP/RTMP: esse passo é necessário para a interoperabilidade entre usuários Flash e não-Flash (ver também Seção 3.3);

- Transcodificação de vídeo: trocar o codec utilizado na *stream* de vídeo. Por exemplo, receber uma *stream* em VP8 e convertê-la para H.264.

Um comando ffmpeg para transcodificação de *streams* tem o seguinte formato (Figura 3.10):

Figura 3.10: Formato do comando *ffmpeg*.

```
ffmpeg -i <stream de entrada> <opções> <stream de saída>
```

Primeiramente, colocamos o endereço da *stream* que queremos transcodificar (*stream* de entrada). Por exemplo, uma *stream* de entrada poderia ser *rtp://143.54.10.53:20881*.

Após, configuramos as opções do comando ffmpeg. Tais opções definem como o ffmpeg será executado e também os parâmetros da transcodificação. Por exemplo, podemos definir se queremos geração de logs, se estes serão impressos em tela ou em arquivo, o nível de detalhe do log que será gerado, entre outros. Já nas opções de transcodificação, descrevemos quais codecs serão utilizados, *bitrates*, resolução, fps, etc (veremos em seguida).

Finalmente, colocamos o endereço para onde a *stream* transcodificada será transmitida. Poderíamos inserir algo como *rtmp://143.54.10.103/video-sala-de-conferencia-3*, caso quiséssemos transcodificar uma *stream* de entrada para RTMP.

A seguir, examinaremos dois exemplos de comandos de transcodificação:

- RTP para RTMP (Figura 3.11):

Figura 3.11: Comando *ffmpeg* para transcodificação de RTP para RTMP.

```
ffmpeg -i rtp://192.168.1.5:20646
-vcodec libopenh264 -profile:v baseline -s 640x480
-r 15 -b:v 1024k -maxrate 1024k
"rtmp://192.168.1.5/video/sala-de-conferencia-3 live=1"
```

1. O primeiro passo do comando é definir o *input*. No caso, estamos utilizando uma *stream* RTP rodando em 192.168.1.5, na porta 20646;

[7] https://ffmpeg.org/ffprobe.html

2. O segundo é estabelecer o codec que será utilizado na transcodificação. No caso, estamos transcodificando a *stream* para H.264 *baseline* (*-vcodec libopenh264 -profile:v baseline*). O comando utiliza a *OpenH264*[8], biblioteca open source e de uso livre. A outra opção seria usar *x264* (ficando: *-vcodec h264 -profile:v baseline*). Essa opção é a biblioteca H.264 da VideoLAN[9], também open source, porém para uso comercial é necessário o pagamento de royalties[10].

   Na configuração do *vcodec*, poderíamos ter utilizado VP8 (*-vcodec vp8*; por consequência, não utilizaríamos *-profile:v baseline*), caso quiséssemos tal conversão.

   Se o desejado fosse não transcodificar o codec, a configuração seria: *-vcodec copy*. Mesmo mantendo o mesmo codec, é possível alterar seus parâmetros, como é apresentado nos próximos itens.

   Nesse exemplo (Figura 3.11), se assume que a *stream* de entrada é uma *stream* de vídeo. Caso tentássemos transcodificar uma *stream* de áudio para H.264 - ou outro codec de vídeo - o comando ffmpeg geraria erro.

3. Após, definimos que o vídeo transcodificado terá a resolução de 640x480 (*-s 640x480*).

4. Em seguida, estabelecemos parâmetros adicionais que serão usados na transcodificação: 15 *fps* (*-r 15*), *bitrate* base de 1024kbps (*-b:v 1024k*), com taxa máxima de, também, 1024kbps (*-maxrate 1024k*); desse modo, definimos que o vídeo gerado será CBR.

5. Finalmente, definimos o endereço para onde será transmitida a *stream* transcodificada (*"rtmp://192.168.1.5/video/sala-de-conferencia-3 live=1"*). Com esse endereço de saída, estamos informando ao ffmpeg que a *stream* convertida para H.264 *baseline* ainda deve sofrer mais uma transcodificação: de RTP para RTMP. A opção *live=1* estabelece que a mídia da *stream* RTMP não tem tamanho final definido (que é o caso das mídias de videoconferência: não há como saber qual o tamanho do áudio e do vídeo que serão transmitidos; a mídia existe enquanto durar a sessão de conferência).

- RTMP para RTP (Figura 3.12):

Figura 3.12: Comando *ffmpeg* para transcodificação de RTMP para RTP.

```
ffmpeg
-i "rtmp://192.168.1.5/video/sala-de-conferencia-3/video-usuario-6 live=1"
-vcodec libopenh264 -profile:v baseline -r 15
-payload_type 97 -b:v 1024k -maxrate 1024k
rtp://192.168.1.5:18888
```

O comando para o caminho contrário é semelhante. Porém, como esperado, a *stream* de entrada é uma *stream* RTMP, e a de saída, RTP. Ainda definimos qual

[8] http://www.openh264.org/
[9] http://www.videolan.org/developers/x264.html
[10] http://x264licensing.com/faq

o valor do *payload type* que os pacotes RTP gerados carregarão (*-payload_type 97*). Como pacotes RTP que transportam vídeo H.264 não tem *payload type* pré-estabelecido (dinâmico), poderíamos ter definido qualquer valor entre 97 e 127 (como explicado na Seção 2.2.1).

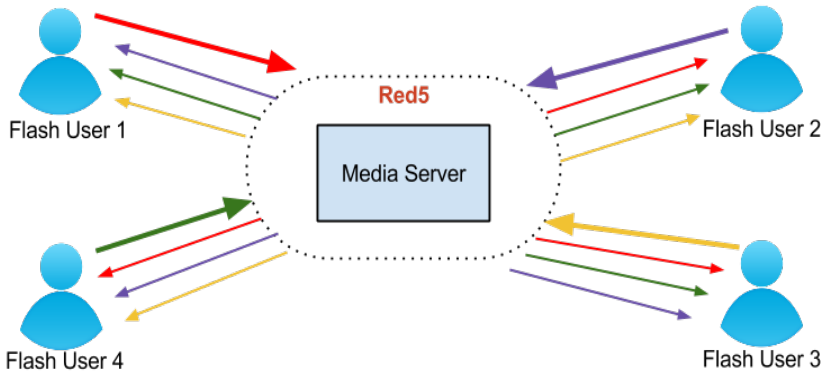
3.3 Red5

O Red5[11] surgiu em 2005 como um servidor de mídia para Flash, e hoje cobre, também, outras tecnologias (como HLS, WebSockets, and RTSP). O Red5 é um meio importante, com a ajuda de outras soluções, para alcançar interoperabilidade de usuários Flash com outros tipos de endpoints. Isso por dois motivos.

O primeiro é justamente ser um servidor de mídia para Flash open source e gratuíto. Sem uma opção desse tipo, depende-se da própria solução da Adobe, o Adobe Media Server, ferramenta paga e fechada.

Cada cliente Flash se conecta ao Red5 através de um *NetConnection*. Uma vez conectados, usuários Flash enviam áudio e vídeo em RTMP. Red5 recebe cada uma das mídias e simplesmente as distribui para os outros clientes Flash (não fazendo mixagem ou *switching*, por exemplo), de forma que cada cliente recebe múltiplas *streams*. A Figura 3.13 ilustra usuários Flash conectados em um servidor Red5, em que cada usuário manda sua mídia (linha colorida em direção ao Red5) e recebe as mídias dos demais usuários (representadas pelas cores de cada usuário).

Figura 3.13: Red5 como servidor de mídia Flash.



O segundo motivo é a possibilidade de adicionar capacidades SIP e de transcodificação RTP/RTMP através do desenvolvimento de aplicações Red5, e, assim, interoperar com endpoints não-Flash. O Red5, nativamente, não suporta SIP. Porém, através da sua API em Java, o Red5 dá suporte a construção de aplicações, e nelas é possível estender as funcionalidades do servidor de mídia. Desse modo, podemos adicionar uma camada SIP. Existem soluções open source já consolidadas como o MjSip[12] e o JAIN-SIP[13], ambas implementadas em Java.

Red5Phone[14] e Bigbluebutton[15], videoconferências open source que utilizam Red5, adicionaram o MjSip como aplicação Red5, incorporando, assim, uma camada SIP ao

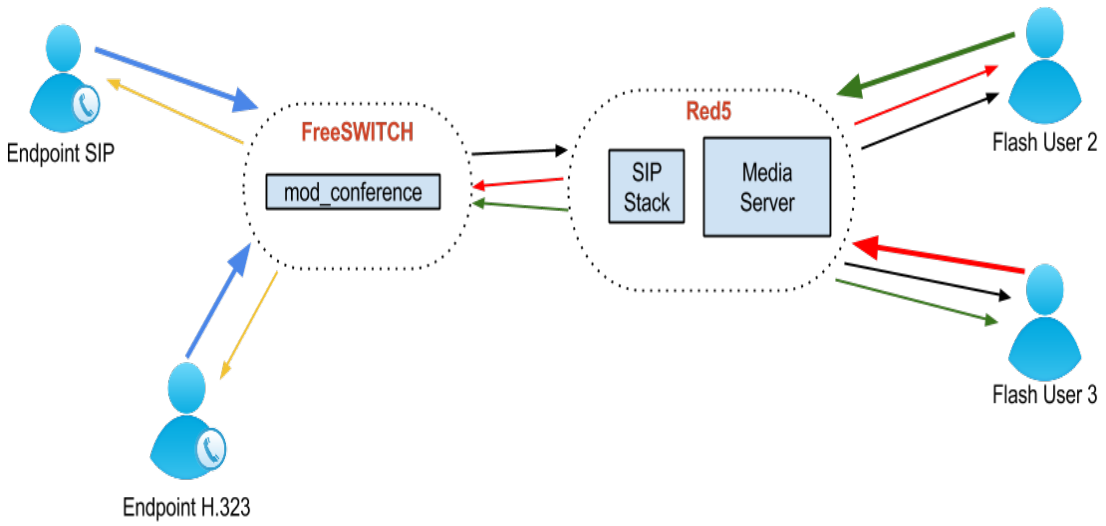
[11] `http://red5.org/`
[12] `http://mjsip.org/index.html`
[13] `https://github.com/RestComm/jain-sip`
[14] `https://code.google.com/archive/p/red5phone/`
[15] `http://bigbluebutton.org/`

servidor de mídia Flash. O recomendado para uma camada SIP de um servidor Red5 é prever um mecanismo de transcodificação RTP/RTMP, visto que a grande maioria de endpoints SIP usam RTP. Os já mencionados Red5Phone e Bigbluebutton possuem um módulo na sua camada SIP que faz essa tradução (códigos próprios, escritos em Java). Alternativamente, poderia se utilizar o FFMPEG, como explicado na Seção 3.2.

Assim, com uma camada SIP executando a transcodificação RTP/RTMP, o Red5 tem a capacidade de se conectar a um softswitch - como o Freeswitch -, transcodificar as mídias dos usuários Flash (RTMP para RTP) e transmiti-las para o softswitch. Este receberia as *streams* e, de acordo com alguma política (por exemplo, mixando os áudios e selecionando o vídeo do falante, como ocorre no Freeswitch), encaminharia a mídia para os usuários SIP/H.323 conectados a ele. Da mesma forma, o softswitch receberia as mídias dos endpoints SIP/H.323 e as enviaria para o Red5, que, por sua vez, executaria a transcodificação de RTP para RTMP e distribuiria o resultado entre os clientes Flash.

A Figura 3.14 ilustra um exemplo de usuários Flash e não-Flash conectados através do Freeswitch e de um servidor Red5. Cada usuário Flash manda sua mídia para o Red5, e este distribui para os demais usuários Flash e também para o Freeswitch. Este, por sua vez, processa as mídias recebidas (mixagem ou *switching*) e transmite o resultado para os *endpoints* (flechas em amarelo). Também há o caminho contrário: os *endpoints* enviam suas mídias para o Freeswitch, este as processa e transmite o resultado do processamento para o Red5 (flecha em preto) que, por sua vez, o distribui entre os usuários Flash. Lembrando que a troca de mídia entre Red5 e Freeswitch envolve transcodificação RTP/RTMP.

Figura 3.14: Red5 e Freeswitch conectando múltiplos usuários.



3.4 Outras Soluções para Trabalhos Futuros

Foram levantadas algumas outras soluções open source que não serão detalhadas, ficando como sugestão a pesquisa destas para futuros trabalhos sobre interoperabilidade. Tais soluções são:

- **Asterisk**[16]: pode atuar como um *softswitch*, sendo uma alternativa ao Freeswitch.

[16] http://www.asterisk.org/

Em testes realizados, foi possível conectar *endpoints* SIP e H.323 via áudio;

- **Kurento**[17]: servidor de mídia WebRTC, onde clientes web se conectam via *WebSocket*. Portanto, não é necessário o uso de bibliotecas como o SIP.js no lado do cliente. Diferentemente do Red5, o Kurento dá opções para a distribuição da mídia, onde o administrador do servidor pode habilitar o uso de mixagem de áudio ou de composição de vídeo. Assim como o Red5, é possível estender as funcionalidades do servidor de mídia através do desenvolvimento de aplicações Kurento. Dessa forma, pode-se adicionar uma camada SIP para a comunicação com *softswitches* ou outros sistemas de videoconferência.

[17] `https://www.kurento.org/`

4 ESTUDO DE CASO: MCONF

Como vimos no capítulo 3, as soluções de interoperabilidade apresentadas implementam apenas parcialmente os padrões de videoconferência. Empiricamente, por mais que uma única solução cubra boa parte desses padrões (como o Freeswitch, por exemplo), o desenvolvimento de um sistema de conferência terá que integrar múltiplas soluções para um resultado final mais abrangente.

Este capítulo, portanto, apresentará uma análise da arquitetura do sistema de web conferência open source Mconf[1] e a modificará para alcançar um nível maior de interoperabilidade, utilizando-se das soluções estudadas no capítulo anterior.

4.1 Mconf

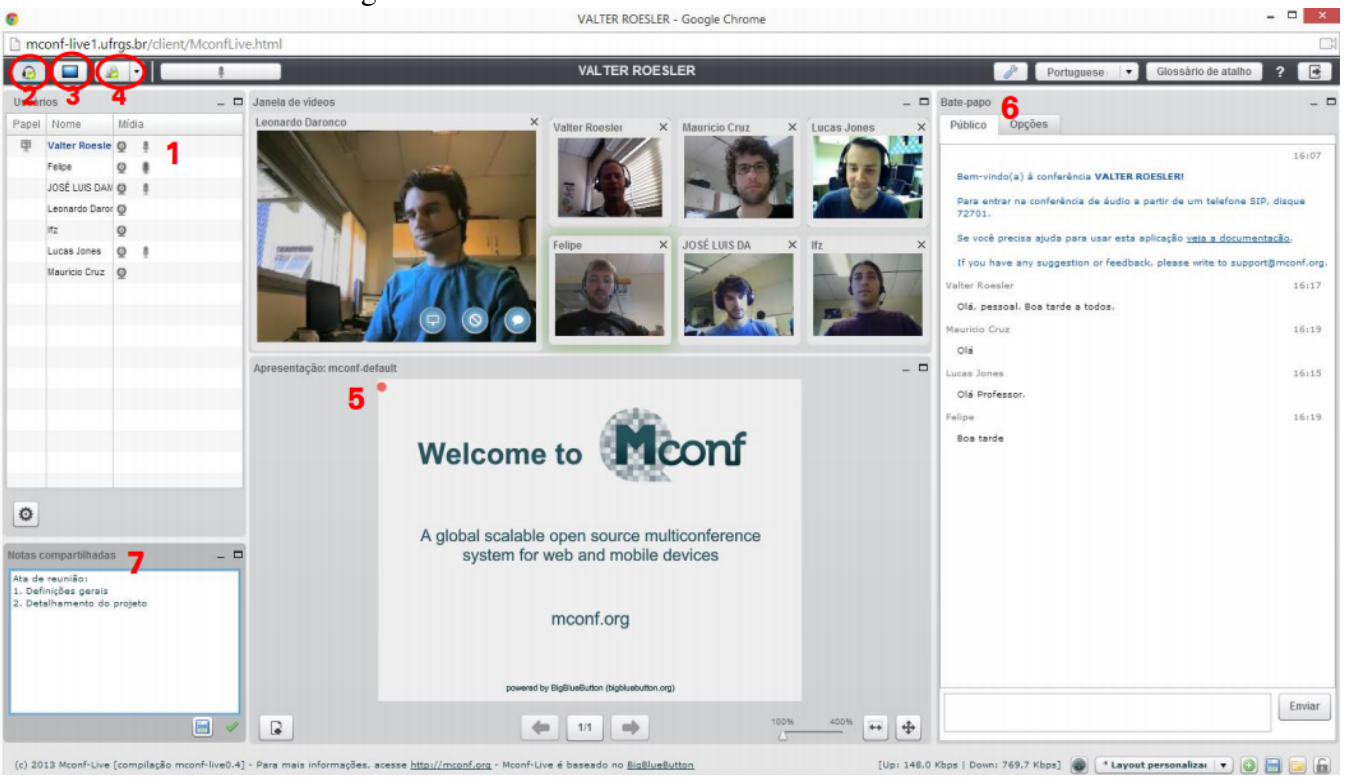
Mconf é um sistema de web conferência open source baseado no BigBlueButton. Basicamente, é composto por dois blocos principais. O primeiro é o *Mconf-Live*, componente de conferência do sistema. Mconf-Live é uma versão personalizada do Bigbluebutton, incluindo novas características, e será o foco deste capítulo, pois nele é definido o Plano de Controle e de Mídia do Mconf. O segundo é o *Mconf-Web*, portal web onde os usuários podem colaborar entre si, agendando conferências.

A Figura 4.1 ilustra a interface web de um usuário conectado em uma sala de conferência do Mconf e, dentre as funcionalidades disponibilizadas, enumera:

1. Lista de participantes: nessa área da interface é mostrada a lista dos usuários conectados à sala. Há 3 tipos de usuários:
    - Moderador: tem poder para expulsar participantes, desligar ou colocar em mudo as mídias ativas, definir qual usuário é o apresentador de sessão, entre outros;
    - Apresentador: tem poder para fazer o upload de slides, usar quadro branco e compartilhar a tela do seu computador. É o tipo de usuário central para reuniões do tipo aula ou palestra;
    - Participante ou *Viewer*: é o usuário comum. Pode utilizar áudio, vídeo, chat e bloco de notas.
2. Botão de ligar áudio: ao clicá-lo, o usuário se conecta no áudio, podendo falar com os outros participantes também conectados. Também é dada a opção de entrar como *Listen Only*, onde o participante apenas ouvirá o áudio dos demais usuários, sem alocação de microfone;

[1] `http://mconf.org/`

Figura 4.1: Interface web do Mconf.



Fonte: (MCONF, 2016).

3. Botão de compartilhamento de tela: ao clicá-lo, o apresentador da sessão passa a transmitir o vídeo da tela do seu computador;

4. Botão de compartilhar câmera: ao clicá-lo, o usuário se conecta no vídeo. Ao conectar-se, é criada uma janela onde é mostrado o vídeo do participante. O conjunto das janelas de todos os participantes compartilhando a câmera, no caso da Figura 4.1, está na parte superior central;

5. Área de apresentação de slides: área onde o apresentador pode fazer o upload e controlar o avanço dos slides. Além disso, é possível fazer anotações e desenhos em cada slide, através do quadro branco;

6. Área de chat: nessa área há o bate papo público, onde todos os usuários visualizam seu conteúdo. Também pode haver o bate papo privado entre 2 usuários;

7. Bloco de notas: local onde todos os usuários podem fazer anotações durante a sessão. Seu conteúdo é compartilhado por todos os participantes.

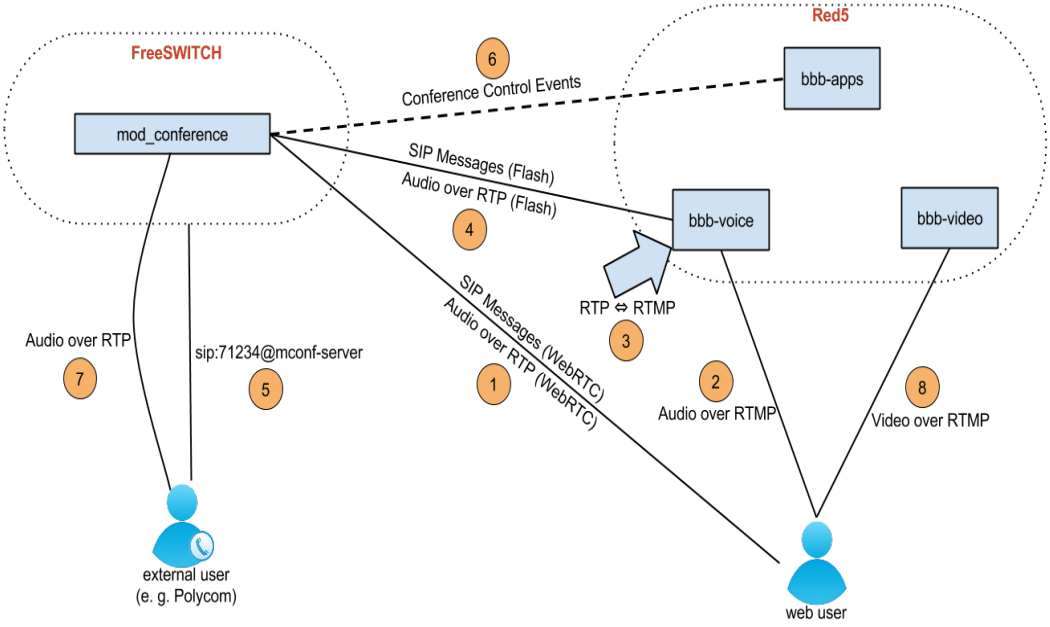
4.1.1 Arquitetura Mconf-Live

Mconf-Live é, por sua vez, formada por dois componentes principais. O primeiro é o Freeswitch, responsável por conectar, através do *mod_conference*, os usuários que estão utilizando áudio. O segundo é o Red5 que, em termos de áudio e vídeo, apresenta duas principais funções. A primeira é receber os áudios dos usuários web utilizando Flash e reencaminhá-los ao Freeswitch. Este fará a mixagem dos áudios e transmitirá de volta as *streams* resultantes para o Red5, que, por sua vez, as distribuirá para os usuários Flash. Dessa forma, cada usuário Flash recebe apenas uma *stream* de áudio, pois, caso

deixássemos a distribuição deste a cargo do Red5, esse tipo de usuário receberia múltiplas *streams* (como explicado na Seção 3.3, na Figura 3.13). A segunda função do Red5 é receber os vídeos dos usuários web e apenas redirecioná-los para os demais usuários. Essa distribuição não envolve o Freeswitch, portanto, se múltiplos usuários web estão enviando vídeo, cada participante receberá múltiplas *streams* de vídeo.

A visão geral da arquitetura do Mconf-Live é demonstrada na Figura 4.2.

Figura 4.2: Visão geral da arquitetura do Mconf-Live.



Primeiramente, analisaremos a arquitetura sob a ótica do usuário web. Mconf-Live, assim como o Bigbluebutton, é baseado em Flash, porém também suporta áudio WebRTC. O comportamento *default* é a conexão de áudio ser via WebRTC, entretanto, se ocorrer algum problema no estabelecimento da chamada, a conexão é feita via Flash.

Mconf-Live utiliza-se do SIP.js para adicionar uma camada SIP ao WebRTC. Assim, a troca de mensagens SIP e o envio/recebimento de áudio RTP ocorrem diretamente entre o navegador web e o Freeswitch (item 1 na Figura 4.2).

Já a conexão de áudio via Flash envolve o Red5. O cliente Flash primeiro se conecta na aplicação Red5 chamada *bbb-voice*, onde está implementada a camada SIP, baseada no MjSip. *bbb-voice* cria um User Agent e envia um INVITE para o Freeswitch. Assim que a chamada for estabelecida, o usuário web passa a enviar áudio RTMP para o User Agent do Red5, que o transcodificará para RTP e o redirecionará para o Freeswitch. Da mesma forma, o User Agent recebe o áudio RTP do FS e, após a transcodificação para RTMP, o reencaminha para o usuário web. Com esse parágrafo, portanto, cobrimos os itens 2, 3 e 4 da Figura 4.2.

Para cada sala do Mconf existe um número de 5 dígitos associado no Freeswitch. Assim, usuários externos (operando algum outro sistema de videoconferência) podem utilizá-lo para efetuar uma ligação SIP (item 5) e, no caso do Mconf-Live, interagir via áudio (apenas) com os usuários web. Assim que uma chamada SIP é estabelecida entre o usuário externo e o Freeswitch, há duas ações que ocorrem concorrentemente.

A primeira ação parte do Freeswitch para o Red5 (item 6): o Freeswitch avisa, via evento XML, que um usuário externo entrou na conferência de áudio. A aplicação Red5

que se comunica via eventos com o Freeswitch é o *bbb-apps*. No momento em que o *bbb-apps* recebe tal evento, ele é responsável por criar um usuário web artificial, incluindo o usuário externo na lista de participantes do contexto web. Dessa forma, os demais usuários web podem visualizar na interface do navegador o usuário externo como participante. Ações de moderador, como colocar um participante em mudo ou desligar seu áudio, são exemplos de ações que envolvem troca de eventos XML entre Red5 e Freeswitch. No primeiro caso, por exemplo, o Red5 envia o evento de *mute user* para o Freeswitch, que passará a ignorar o áudio desse usuário na mixagem.

A segunda ação após o estabelecimento da chamada é, naturalmente, o usuário externo enviar/receber áudio para/do Freeswitch (item 7). Dessa forma, o Freeswitch consegue conectar, via áudio, usuários WebRTC, usuários Flash (através do User Agent criado no Red5) e também usuários externos. Todos os áudios RTP recebidos são mixados (com eventuais transcodificações de codecs, como explicado na Seção 3.1) e distribuídos entre os usuários conectados, sem nenhuma distinção de qual tecnologia, hardware ou software o usuário está utilizando.

O Mconf-Live ainda tem uma terceira aplicação Red5, o *bbb-video*. Através dela, ocorre a distribuição dos vídeos RTMP dos usuários web. Como explicado no começo dessa subseção, a conexão do vídeo não envolve o Freeswitch, de forma que apenas usuários web podem enviar e receber vídeo. A conexão do vídeo é independente da de áudio, portanto o usuário compartilhando a câmera pode estar com o áudio ligado via WebRTC ou Flash, ou com o áudio desligado. Como a distribuição do vídeo ocorre apenas através do Red5, cada usuário web envia sua *stream* de vídeo RTMP (item 8) e recebe de volta as *n streams* dos demais usuários web compartilhando a câmera (como na Figura 3.13, do capítulo anterior).

4.2 Estendendo a interoperabilidade do Mconf

Nessa seção, adicionaremos no Mconf duas novas formas de interoperabilidade com usuários externos. Na primeira, possibilitaremos que equipamentos ou softwares H.323 liguem para uma sala do Mconf e interajam via áudio. Na segunda, daremos suporte para que usuários externos SIP possam interagir com os usuários web não só via áudio, mas também via vídeo.

4.2.1 Suporte a H.323

Para que usuários externos H.323 possam se conectar em salas do Mconf, deve-se habilitar o *mod_h323* no Freeswitch. Uma vez habilitado, o usuário H.323, ao ligar algo como *h323:72013@servidor-do-mconf.com*, trocará mensagens H.323 com o *mod_h323*, e este passará o destino (no caso, o destino é a sala 72013) para o módulo de *dialplans* do Freeswitch. Ao encontrar o dialplan que, checando que o destino tem 5 dígitos, encaminha a ligação para a sala 72013 do *mod_conference* (como foi exemplificado na Seção 3.1, especialmente em 3.1.1), o usuário H.323 estará finalmente conectado.

São necessários os seguintes passos para habilitar o *mod_h323*:

1. Instalar as bibliotecas *PTlib* e *H323 Plus*: o *mod_h323* depende da API da biblioteca open source H.323 Plus[2] para o seu funcionamento, que, por sua vez, depende da biblioteca *PTlib*. É necessário atentar às versões de ambas as bibliotecas, pois há versões do H.323 Plus que não são compatíveis com certas versões do PTLib. A

[2] `http://www.h323plus.org/source/`

combinação funcional mais recente à época deste trabalho é: H.323 Plus 1.26.5 e PTLib 2.12.8.

2. Compilar o *mod_h323*: Após a instalação das bibliotecas, deve-se baixar o código fonte do Freeswitch e compilar o *mod_h323*, já que este não é um módulo padrão e, portanto, não faz parte do pacote de instalação do FS. Por exemplo, uma compilação no Ubuntu ocorreria da seguinte maneira (Figura 4.3):

Figura 4.3: Compilação do *mod_h323* no Ubuntu.

```
cd pasta-do-codigo-fonte-do-freeswitch
./bootstrap.sh
./configure –prefix=/opt/freeswitch
make mod_h323
make mod_h323-install
```

3. Criar o arquivo *h323.conf.xml*: Esse arquivo define uma série de configurações que serão usadas pelo Freeswitch nas ligações H.323. Tal arquivo deve ser criado em *conf/autoload_configs/*, localizado dentro da pasta onde o Freeswitch está instalado (no exemplo do item 2, o Freeswitch está instalado em */opt/freeswitch*). Um exemplo de *h323.conf.xml* é ilustrado na Figura 4.4.

Figura 4.4: Exemplo de *h323.conf.xml*.

```
1  <configuration name="h323.conf" description="H323 Endpoints">
2   <settings>
3     <param name="trace-level" value="6"/>
4     <param name="context" value="public"/>
5     <param name="codec-prefs" value="PCMA,PCMU,GSM,G729"/>
6   </settings>
7   <listeners>
8     <listener name="default">
9       <param name="h323-ip" value="$${local_ip_v4}"/>
10      <param name="h323-port" value="1720"/>
11    </listener>
12  </listeners>
13 </configuration>
```

Na Figura 4.4 são configurados o *trace-level*, número de 1 a 7, que define o nível de detalhamento dos logs gerados pelo *mod_h323*; o *context*, que descreve o subconjunto de dialplans que serão examinados em ligações H.323 (no caso, todos os *dialplans* da pasta *conf/dialplans/public*); e o *codec-prefs*, que estabelece quais codecs o Freeswitch aceitará nas negociações de mídia H.245. Ainda estabelece o IP e a porta do canal de sinalização H.323 (*h323-ip* e *h323-port*). No caso do *h323-ip*, está sendo usado o IP contido na variável *local_ip_v4*, que é definido em *vars.xml*.

4. O último passo é tornar o *mod_h323* carregável ao iniciar o Freeswitch. Para tal, deve-se adicionar a linha *<load module="mod_h323"/>* em *conf/autoload_configs/modules.conf.xml* e, após, reiniciar o Freeswitch.

Um ponto negativo do *mod_h323* é apenas aceitar ligações H.323 de áudio. Isso pois o campo *codec_prefs* em *h323.conf.xml* só aceita os seguintes codecs: PCMA, PCMU, GSM, G723, G729B, G729, G729A, G729A/B, G723.1.

Após habilitar o *mod_h323*, usuários externos H.323 podem se conectar a uma sala de conferência do Mconf e interagir com os participantes via áudio, da mesma forma que usuários externos SIP o fazem atualmente.

Para mais informações sobre o módulo H.323 do Freeswitch, é possível consultar a página web dedicada ao *mod_h323* em *freeswitch.org*[3].

4.2.2 Suporte a Vídeo SIP

Já para dar suporte à interoperabilidade de usuários externos SIP via áudio e vídeo, foram necessárias mudanças não só nas configurações dos componentes, mas também na estrutura da arquitetura do Mconf-Live.

A decisão de projeto foi continuar usando o Red5 para a distribuição do vídeo. Entretanto, o *bbb-video* receberia, além de todos os vídeos RTMP dos usuários web, mais uma *stream* de vídeo: o vídeo do Freeswitch, resultante do processo de *switching*. Assim, quando um usuário externo SIP se tornasse o *video floor*, o vídeo extra no *bbb-video* seria distribuído para todos os usuários web, que veriam, então, o vídeo desse usuário externo.

Para isso, o Freeswitch agora teria de receber o vídeo de todos os participantes (web e externos). Recebendo-os, o Freeswitch tem condição de avaliar o usuário falante e enviar o vídeo correspondente via *switching*.

Todavia, um problema no que diz respeito à usabilidade teve de ser resolvido: o caso do usuário web ser o *video floor*. No contexto do usuário externo não havia problema, pois, durante a fala do usuário web, o Freeswitch selecionaria o vídeo desse e o enviaria para todos os participantes conectados, e, então, usuários externos SIP veriam a imagem do usuário web. O problema era o contexto web. O *bbb-video*, como já esclarecido, recebe e distribui todos os vídeos dos usuários web, de forma que cada usuário desse tipo vê, em sua interface gráfica no navegador, os vídeos dos demais usuários web compartilhando a câmera. Nesse sentido, caso o *video floor* seja justamente um usuário web, o *bbb-video* - que agora também recebe e distribui o vídeo resultante do *switching* - receberia e distribuiria o mesmo vídeo duas vezes, fazendo a interface do contexto web mostrar duas janelas com o mesmo conteúdo de imagem. A decisão, portanto, foi não exibir, no navegador web, o *switching* do Freeswitch quando este envia um vídeo de usuário web, já que esse já estaria sendo exibido (a ilustração desse caso está no próximo capítulo, na Seção 5.1.2).

Para implementar a interoperabilidade de áudio e vídeo com usuários externos SIP, foram necessárias as seguintes mudanças:

- **Habilitar chamadas SIP de vídeo no Freeswitch**: A configuração atual do Freeswitch do Mconf-Live implica que toda chamada SIP será apenas de áudio, mesmo se o endpoint SIP que efetuou a chamada avisar, via SDP, que suporta vídeo. Para aceitarmos ligações SIP desse tipo, devemos habilitar pelo menos algum codec de vídeo na variável *global_codec_prefs* em *vars.xml*, local onde são listados os codecs permitidos nas negociações de mídia. A decisão do projeto foi habilitar apenas H.264, visto que esse é o codec de vídeo de alta qualidade mais comum em sistemas de videoconferência;

- **Listar vídeo como mídia suportada no SDP dos usuários web**: quando um usuário web se conecta no áudio, o SDP que o Freeswitch recebe (seja do SIP.js - no

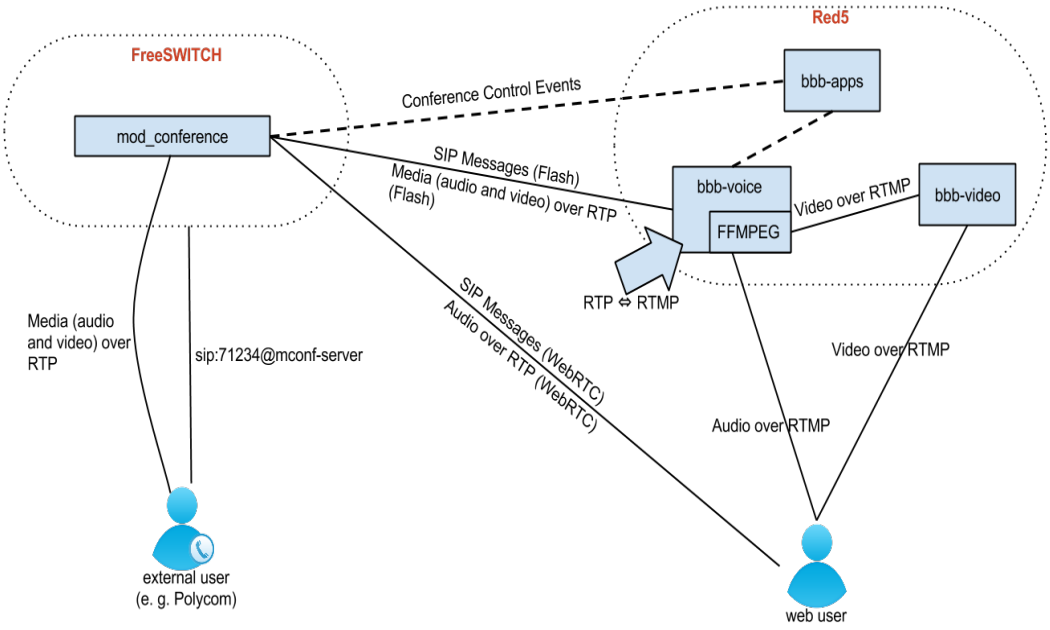
[3] `https://freeswitch.org/confluence/display/FREESWITCH/mod_h323`

caso de áudio WebRTC -, ou do User Agent criado no Red5 - se áudio Flash) contém apenas informações de áudio. Para o Freeswitch receber os vídeos do contexto web, é necessário estabelecer uma chamada SIP de áudio e vídeo e, para isso, modificamos a criação de SDPs tanto do SIP.js quanto do *bbb-voice*, a fim de incluir vídeo como mídia suportada, adicionando, naturalmente, H.264 como seu atributo;

- **Adição do módulo FFMPEG em *bbb-voice***: O módulo *bbb-voice* possui, em seu código fonte, um mecanismo de transcodificação RTP/RTMP para áudio, que poderia ser estendido também para vídeo. Entretanto, essa seria uma tarefa trabalhosa, visto que a transcodificação RTP/RTMP de *streams* de vídeo apresenta complexidade maior que a de *streams* de áudio, de forma que foi decidido integrar um submódulo FFMPEG ao *bbb-voice*. Esse submódulo, portanto, teria as seguintes funções:
  - Rodar sobre as *streams* RTMP do *bbb-video*, transcodificá-las para RTP e enviá-las para o Freeswitch. Tal transmissão usaria IP e portas de vídeo que foram definidos nas negociações SIP, que ocorreram quando os usuários web se conectaram no áudio (seja via WebRTC ou Flash). Na prática, transcodificaremos apenas uma *stream* RTMP, como explicaremos mais adiante;
  - Rodar sobre a *stream* de vídeo RTP que o Freeswitch gera a partir do *switching*, transcodificá-la para RTMP e publicá-la no *bbb-video*, a fim da distribuição entre os usuários web. Se o vídeo do *switching* for de algum usuário web, não há a publicação no *bbb-video*. Portanto, só ocorre a distribuição do vídeo do Freeswitch entre os usuários web quando tal vídeo é de um usuário externo.
- **Administrar novos eventos XML no bbb-apps**: Em um cenário onde existem usuários conectados com áudio e vídeo no *mod_conference*, este dispara um evento XML para seus *subscribers* - que é o caso do *bbb-apps* - toda vez que há um novo *video floor*. Ao receber tal evento, o *bbb-apps* passa a saber quem é o *video floor* atual (pois um dos argumentos do evento é o identificador desse) e repassa essa informação ao *bbb-voice*. Este, por sua vez, toma as seguintes decisões:
  - Se o *video floor* for um usuário web, transcodificaremos apenas a sua *stream* de RTMP para RTP. Dessa forma, não transmitimos todas as *streams* do *bbb-video* para o Freeswitch, poupando uso de CPU no servidor (já que não haverá múltiplas transcodificações de vídeo). Além disso, identificando que o *video floor* é um usuário web, sabemos que não precisamos publicar o resultado do *switching* no *bbb-video*.
  - Se o *video floor* for um usuário externo, o módulo FFMPEG irá transcodificar a *stream* RTP do Freeswitch para RTMP e publicá-la no *bbb-video*.

A visão geral da nova arquitetura pode ser conferida na Figura 4.5.

Figura 4.5: Visão geral da arquitetura modificada do Mconf-Live.



Com a arquitetura modificada, já temos condições de testar a nova interoperabilidade do Mconf com softwares e equipamentos de videoconferência.

5 RESULTADOS

Os testes de interoperabilidade com o Mconf validaram os seguintes sistemas de videoconferência:

- **Ekiga**[1]: Lançado em 2001 com o nome de GnomeMeeting, Ekiga é um softfone SIP e H.323 open source, implementando diversos codecs de áudio, como Speex, G.7xx e iLBC, e suportando vídeo H.263 (as 3 versões deste) e H.264.

- **Jitsi**[2]: antes conhecido como JsPhone, Jitsi é um softfone SIP open source desenvolvido em Java, e cuja camada SIP é baseada na biblioteca JAIN-SIP. O projeto Jitsi tem focado no desenvolvimento de produtos WebRTC, como o Jitsi Meet[3], sistema de webconferência, e o Jitsi Videobridge[4], hardware dedicado para conferência de múltiplos usuários, também compatível, naturalmente, com WebRTC. Para áudio, além de Speex, G.7xx e iLBC, Jitsi também implementa Opus. Já para vídeo, há suporte para H.263-1998, H.264 e VP8;

- **Polycom QDX 6000**[5]: QDX 6000 é um sistema de videoconferência de sala desenvolvido pela Polycom[6], com suporte a SIP, vídeo H.263 e H.264, e diversos codecs de áudio. Para a validação da interoperabilidade com o Mconf, foi utilizado a unidade da Sala de Videoconferência do Instituto de Informática da UFRGS.

- **Huawei TE-30**[7]: TE-30 é um sistema de videoconferência de sala SIP e H.323, desenvolvido pela Huawei[8]. Suporta G.7xx, H.263 e H.264.

- **Polycom RMX 2000**[9]: RMX 2000 é um MCU desenvolvido pela Polycom, conectando múltiplos usuários SIP e H.323, mixando áudio, compondo vídeo e distribuindo as mídias entre os participantes. Implementa áudio G.7xx e vídeo H.26x.

---

[1] http://www.ekiga.org/

[2] http://www.jitsi.org/

[3] https://jitsi.org/Projects/JitsiMeet

[4] https://jitsi.org/Projects/JitsiVideobridge

[5] http://support.polycom.com/PolycomService/support/us/support/video/qdx/qdx_6000.html

[6] http://www.polycom.com.br/

[7] http://e.huawei.com/br/products/enterprise-networking/telepresence-video-conferencing/endpoints/te-30

[8] http://e.huawei.com/

[9] http://support.polycom.com/PolycomService/support/us/support/network/collaboration_conferencing_platforms/rmx_2000.html

- **Polycom ViewStation FX**[10]: Outro sistema de videoconferência de sala da Polycom, com suporte apenas a H.323. Implementa G.722, G.728 e H.26x.

Ambos sistemas implementam o padrão RTP para o transporte de mídia.

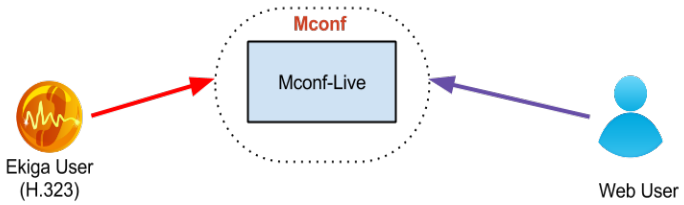
5.1 Casos de Uso

Nessa seção, serão apresentados alguns casos de uso dos testes de interoperabilidade. Cada usuário externo (operando algum dos sistemas listados acima) acessou a sala 72013 do servidor Mconf-Live hospedado em *lab-mconf-live-sip.rnp.br*.

5.1.1 Caso 1

A Figura 5.1 ilustra o Caso de Uso 1, demonstrando um uso básico de interoperabilidade H.323.

Figura 5.1: Caso de Uso 1.



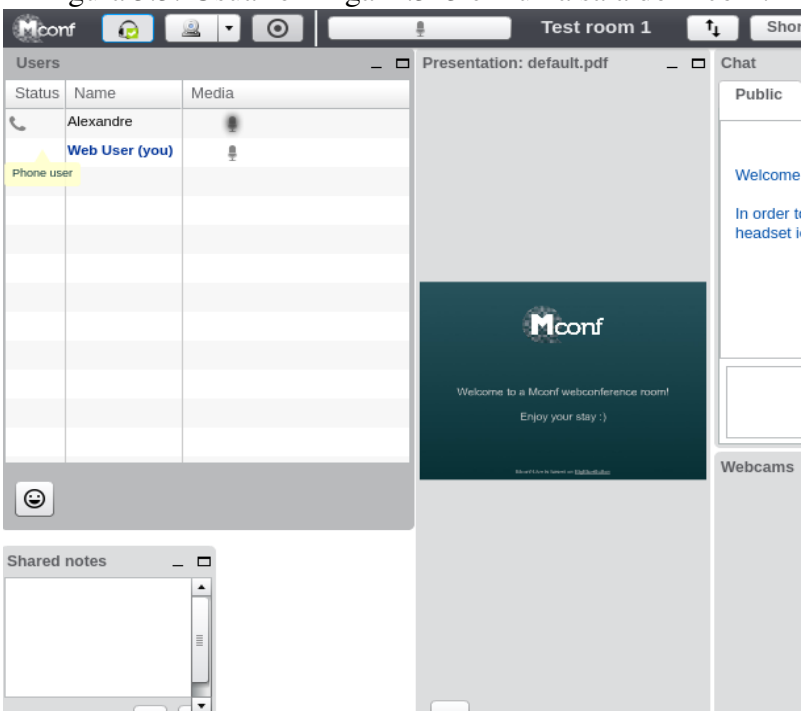
O acesso à sala 72013 via H.323 pelo usuário Ekiga, com o *username* 'Alexandre', se dá através da ligação *h323:72013@lab-mconf-live-sip.rnp.br*, como mostra a Figura 5.2.

Figura 5.2: Ligação H.323 Ekiga.

[10] http://support.polycom.com/PolycomService/support/us/support/video/other_video/viewstation_fx.html

Assim, ao efetuar a chamada, o usuário Ekiga será um dos integrantes da sala de conferência, sendo exibido na lista de participantes na interface do navegador web, e poderá interagir via áudio com o usuário web '*Web User*', como demonstrado na Figura 5.3. Ainda na interface, indicamos com um pequeno símbolo de telefone à esquerda que o usuário em questão é um usuário externo. No momento da captura da Figura 5.3, o usuário Ekiga H.323 estava falando (o que implica no brilho do ícone de microfone à direita do *username* 'Alexandre').

Figura 5.3: Usuário Ekiga H.323 em uma sala do Mconf.

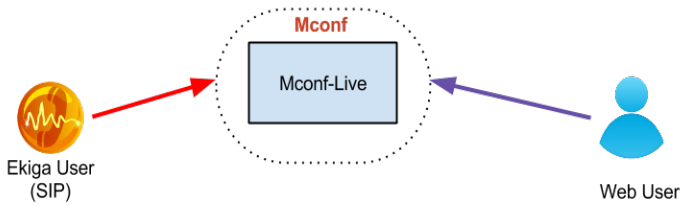


Tal caso de uso foi repetido com os demais softwares e equipamentos H.323, como o Huawei TE-30 e o Polycom ViewStation FX, onde ambos interoperaram com sucesso.

5.1.2 Caso 2

A Figura 5.4 ilustra o Caso de Uso 2, demonstrando um uso básico de interoperabilidade SIP.

Figura 5.4: Caso de Uso 2.

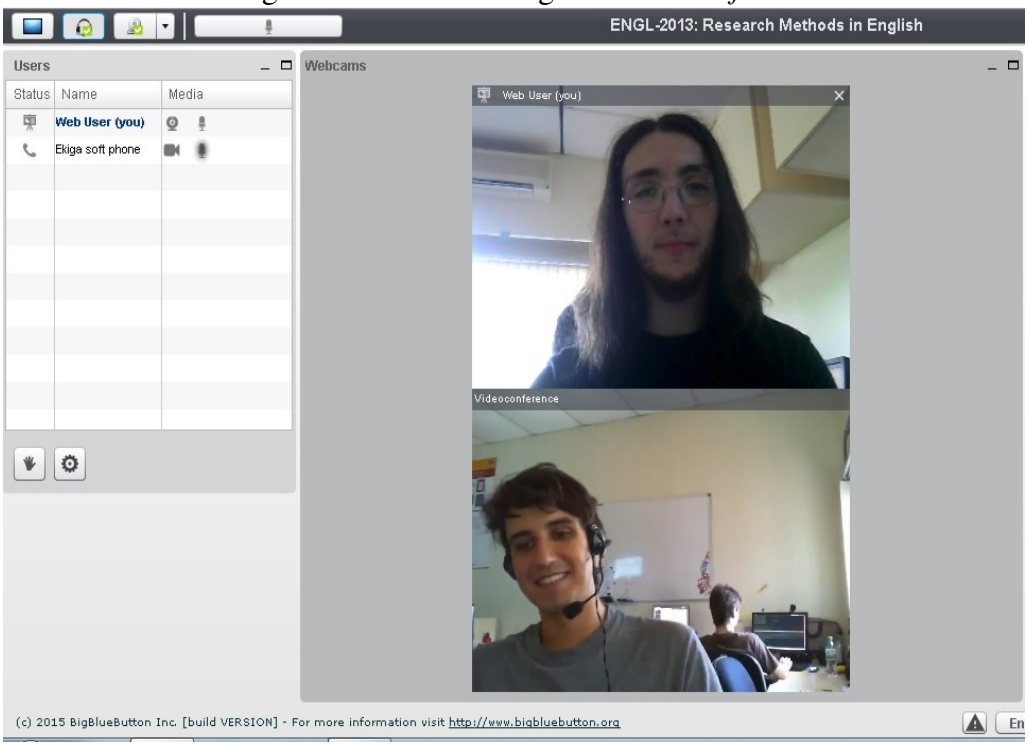


O acesso à sala 72013 via SIP pelo usuário Ekiga, com o *username* 'Ekiga soft phone', se dá através da ligação *sip:72013@lab-mconf-live-sip.rnp.br*. As Figuras 5.5 e 5.6 de-

monstram como a interoperabilidade SIP via áudio e vídeo se reflete na interface web do Mconf.

Na primeira figura (5.5), o *video floor* é o usuário Ekiga (como mencionado, o ícone de microfone brilhando indica que esse usuário está falando). Assim, o resultado do *switching* do Freeswitch transmite o vídeo do usuário Ekiga para o *bbb-voice*, que, checando que tal vídeo é de um usuário externo, publica-o no *bbb-video* para ser distribuído entre os usuários web. A janela do cliente web responsável por exibir vídeos de usuários externos é a janela *Videoconference* (posicionada mais abaixo na Figura 5.5). No momento em que o usuário Ekiga se torna o *video floor*, ele também recebe o próprio vídeo (como explicado na Seção 3.1.1).

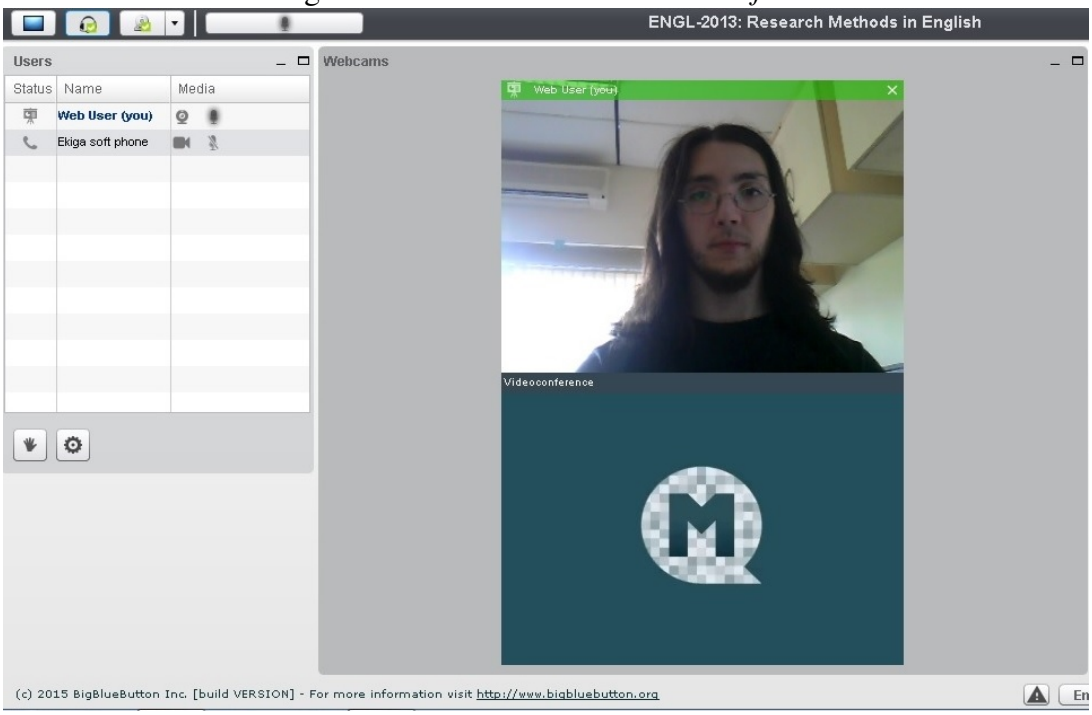
Figura 5.5: Usuário Ekiga como *video floor*.



Já quando o *Web User* começa a falar, o *bbb-voice* - que receberá o vídeo desse usuário como resultado do *switching* - verificará que foi um usuário web que se tornou o *video floor*, e não publicará o vídeo do Freeswitch no *bbb-video*. Desse modo, a janela *Videoconference* exibe um gif animado para não haver vídeo duplicado, como pode ser conferido na Figura 5.6. Já o usuário externo receberá e exibirá o vídeo do *Web User* normalmente.

Tal caso de uso também foi repetido com os demais softwares e equipamentos SIP, validando a interoperabilidade com o Jitsi, Huawei TE-30 e Polycom QDX 6000, onde ambos interoperaram com sucesso.

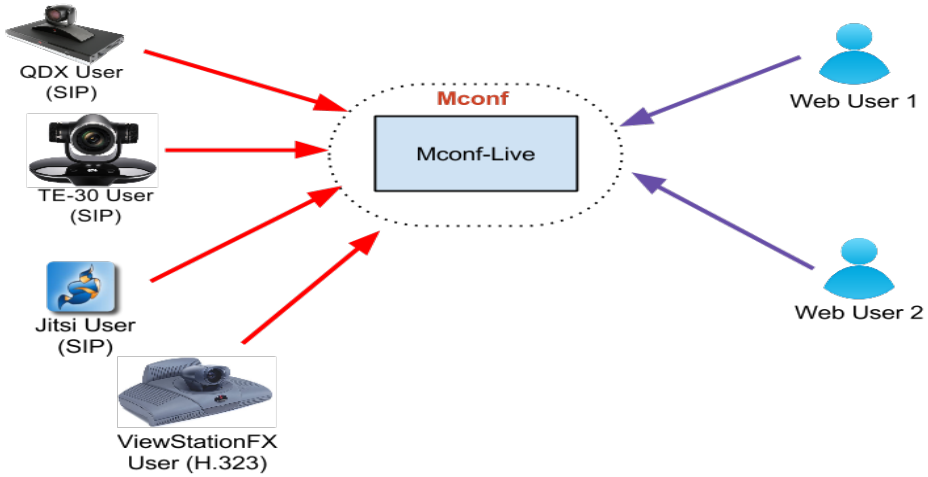
Figura 5.6: Usuário web como *video floor*.



5.1.3 Caso 3

A Figura 5.7 ilustra o Caso de Uso 3, validando a interoperabilidade do Mconf com múltiplos tipos de usuários simultâneos. Nesse teste, o Mconf-Live conecta dois usuários web e quatro sistemas externos: Polycom QDX 6000, Huawei TE-30, Jitsi e Polycom ViewStation FX (este está apenas via áudio, pois só utiliza H.323).

Figura 5.7: Caso de Uso 3.



A Figura 5.8 apresenta a interface do Mconf, onde há as duas janelas de vídeo dos usuários web (*Web User 1* e *Web User 2*), e a janela *Videoconference*, que, no momento, está mostrando o vídeo do Polycom QDX 6000 (*QDX User* é o *video floor*).

Já a Figura 5.9 é uma fotografia do Polycom QDX 6000 da Sala de Videoconferência do Instituto de Informática da UFRGS (o *QDX User* do Caso 3) exibindo o vídeo do *Web User 1*, quando este se tornou o *video floor* em um momento posterior.

Figura 5.8: Mconf exibindo o vídeo do Polycom QDX 6000 (*QDX User*).

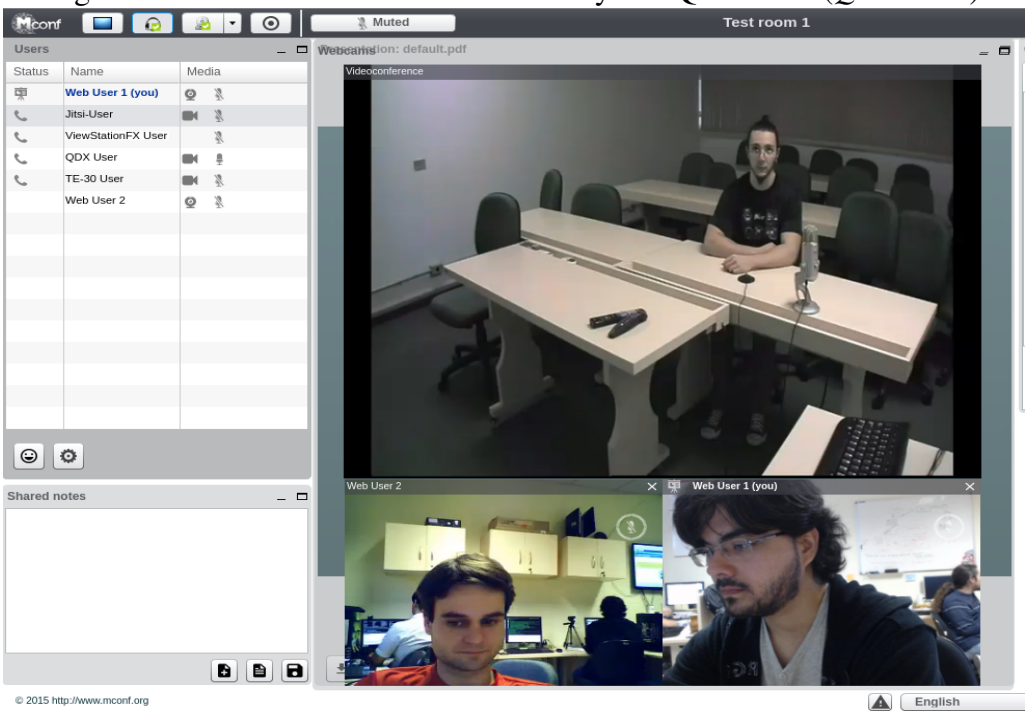


Figura 5.9: Polycom QDX 6000 exibindo vídeo do usuário web *Web User 1*.

5.1.4 Caso 4

A Figura 5.10 ilustra o Caso de Uso 4, validando a interoperabilidade do Mconf com o MCU RMX 2000 da Polycom. Conectaram-se ao MCU 3 sistemas de videoconferência: Ekiga, Huawei TE-30 e Jitsi. Desse modo, quando o MCU se conecta ao Mconf (através de uma ligação SIP), temos na interface web o que está ilustrado na Figura 5.11. Nela, há os vídeos dos usuários web e, no *Videoconference*, a composição de vídeo feita pelo RMX 2000, mostrando todos os usuários externos.

Figura 5.10: Caso de Uso 4.

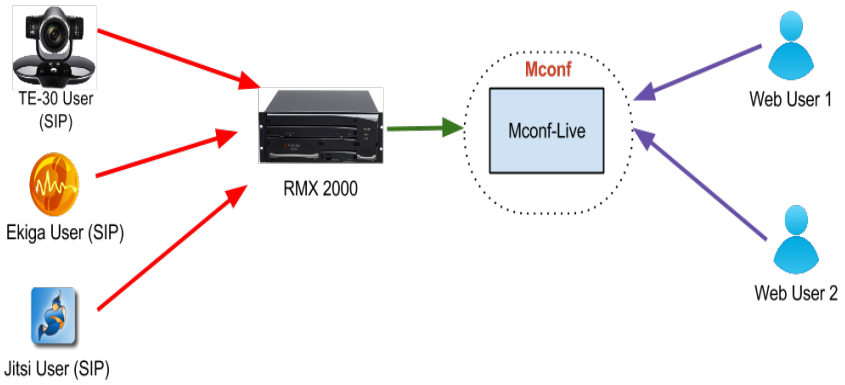
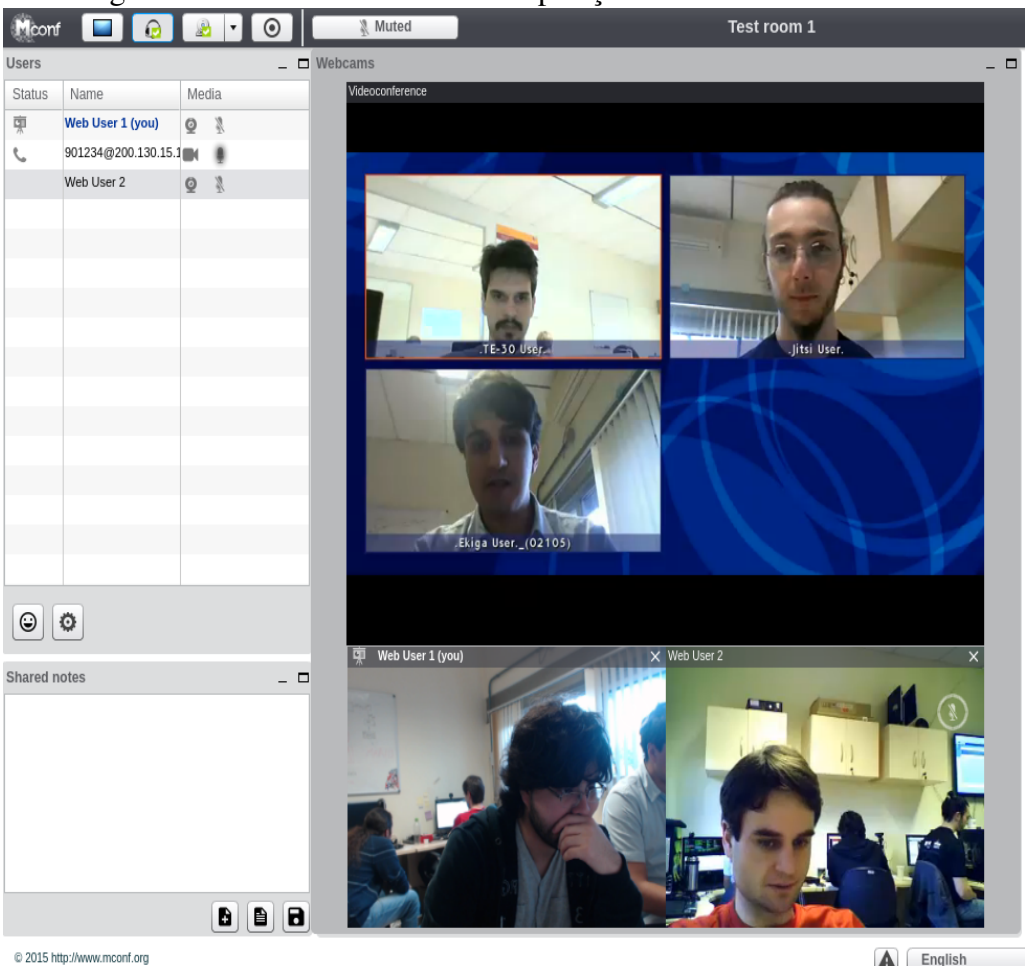


Figura 5.11: Mconf exibindo a composição de vídeo do RMX 2000.



6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Através do estudo dos diversos tipos, padrões e tecnologias de videoconferência (Capítulo 1 e 2), e do levantamento de soluções open source que integram tais padrões (Capítulo 3), este trabalhou apresentou meios que possibilitam sistemas de webconferência interoperarem com os mais diversos softwares e equipamentos de videoconferência, apresentando um estudo de caso onde se estendeu a interoperabilidade do sistema Mconf, utilizando-se das soluções apresentadas.

Por meio do mesmo estudo de caso, é possível, ainda, realizar melhorias ou investigar novas formas de interoperabilidade. Para tal, são apresentadas duas sugestões de trabalhos futuros:

1. **Explorar Freeswitch 1.6**: como já mencionado no Capítulo 3, à época da finalização deste trabalho, foi lançado o Freeswitch versão 1.6, com diversas novas *features* para vídeo, permitindo seleção e composição.

   Através da seleção de vídeos, uma importante melhoria que pode ser desenvolvida é exibir simultaneamente todos os vídeos dos usuários externos na interface web do Mconf. Assim, por exemplo, o *bbb-voice* selecionaria os múltiplos vídeos de usuários externos e os publicaria no *bbb-video*.

   Além disso, com essa função, usuários WebRTC poderiam enviar seu vídeo diretamente para o Freeswitch (assim como acontece com o áudio), e o *bbb-voice* selecionaria, além dos vídeos dos usuários externos, os vídeos dos usuários WebRTC. Vídeos WebRTC geralmente utilizam VP8, portanto, o Freeswitch deve ser reconfigurado para aceitar chamadas que usam tal codec. O submódulo FFMPEG no *bbb-voice* transcodificaria tais vídeos de VP8 para H.264 e os publicaria no *bbb-video*.

   Outra opção seria uma espécie de 'Modo MCU' no Mconf, onde existiria apenas uma janela de vídeo ativa na sala de conferência, exibindo a composição de vídeo de todos os usuários (web e externos). Assim, cada usuário - utilizando o navegador web ou outro sistema de videoconferência - receberia apenas uma *stream* de vídeo. Contudo, composição de vídeo é uma tarefa complexa, e, desse modo, deve-se tomar cuidado com a carga do servidor que está hospedando o Freeswitch.

2. **Explorar Kurento**: O projeto Bigbluebutton aos poucos se encaminha para um cliente web puramente baseado em HTML5. Assim, a distribuição de mídia do contexto web deverá ser feita exclusivamente via WebRTC (que é parte do conjunto de tecnologias que o HTML5 define). Um servidor de mídia WebRTC open source citado neste trabalho é o Kurento, que trabalha de uma maneira similar ao Red5.

Uma sugestão de trabalho seria conectar os usuários web em um servidor Kurento (portanto, o SIP.js não seria mais necessário), e, por meio de uma aplicação Kurento que ofereça uma camada SIP, conectá-lo ao Freeswitch a fim da distribuição de mídia entre usuários externos.

3. **Adicionar capacidades de vídeo no *mod_h323***: Como o Freeswitch é um projeto open source, uma importante contribuição seria dar suporte a algum codec de vídeo no *mod_h323*. Dessa maneira, o *bbb-voice* poderia receber do Freeswitch também a imagem dos usuários H.323 conectados.

REFERÊNCIAS

ADOBE. **Adobe**. [Online; acessado em 26-Maio-2016], http://www.adobe.com/.

Adobe Connect: bringing together video and next-generation web conferencing. **White paper**, [S.l.], 2011.

ASTERISK. **Asterisk**. [Online; acessado em 26-Maio-2016], http://www.asterisk.org/.

BIGBLUEBUTTON. **Bigbluebutton**. [Online; acessado em 22-Maio-2016], http://bigbluebutton.org/.

CROWCROFT, J.; HANDLEY, M.; WAKEMAN, I. **Internetworking Multimedia**. [S.l.]: Morgan Kaufmann, 2009.

EKIGA. **Ekiga**. [Online; acessado em 26-Maio-2016], http://www.ekiga.org/.

FFMPEG. **FFMPEG**. [Online; acessado em 22-Maio-2016], https://ffmpeg.org/.

FIRESTONE, S.; RAMALINGAM, T.; FRY, S. **Voice and Video Conferencing Fundamentals**. [S.l.: s.n.], 2007.

FREESWITCH. **Freeswitch**. [Online; acessado em 22-Maio-2016], https://freeswitch.org/.

HARTPENCE, B. **Packet Guide to Voice over IP**. [S.l.]: O'Reilly Media, Inc., 2013.

HUAWEI. **Huawei**. [Online; acessado em 26-Maio-2016], http://www.huawei.com/.

IETF. **SIP RFC 3261**. [Online; acessado em 4-Maio-2016], https://tools.ietf.org/html/rfc3261.

IETF. **RFC 4587**. [Online; acessado em 8-Maio-2016], https://tools.ietf.org/html/rfc4587.

IETF. **RFC 6386**. [Online; acessado em 9-Maio-2016], https://tools.ietf.org/html/rfc6386.

ITU. **H.225 Recommendation (12/09)**. [Online; acessado em 4-Maio-2016], http://www.itu.int/rec/T-REC-H.225.0-200912-I/en.

ITU. **H.323 Recommendation (12/09)**. [Online; acessado em 4-Maio-2016], http://www.itu.int/rec/T-REC-H.323-200912-I/en.

ITU. **H.245 Recommendation (05/11)**. [Online; acessado em 4-Maio-2016], http://www.itu.int/rec/T-REC-H.245-201105-I/en.

ITU. **H.264 Recommendation (02/16)**. [Online; acessado em 9-Maio-2016], https://www.itu.int/rec/T-REC-H.264-201602-P/en.

JITSI. **Jitsi**. [Online; acessado em 26-Maio-2016], http://www.jitsi.org/.

KARVIR, V. **Delivering Enterprise Applications, Content, and Communications with the Flash Platform**. [S.l.: s.n.], 2005.

KUROSE, J. F.; ROSS, K. W. **Computer Networking**: a top-down approach, 6th edition. [S.l.]: Pearson, 2013.

MCONF. **Mconf**. [Online; acessado em 22-Maio-2016], http://mconf.org/.

OGUNFUNMI, T.; NARASIMHA, M. **Principles of Speech Coding**. [S.l.]: CRC Press, 2010.

PARMAR, H.; THORNBURGH, M. **Adobe's Real Time Messaging Protocol**. [S.l.]: Copyright Adobe Systems Incorporated, 2012.

POLYCOM. **Polycom**. [Online; acessado em 26-Maio-2016], http://www.polycom.com.br/.

RED5. **Red5**. [Online; acessado em 22-Maio-2016], http://red5.org/.

ROESLER, V. et al. Mconf: an open source multiconference system for web and mobile devices. In: **Multimedia - A Multidisciplinary Approach to Complex Issues**. [S.l.]: InTech, 2012.

SCHULZRINNE, H. et al. **RTP**: a transport protocol for real time applications. [S.l.: s.n.], 2003.

SKYPE. **Skype**. [Online; acessado em 26-Maio-2016], http://www.skype.com/.

TANENBAUM, A. S.; WETHERALL, D. J. **Computer Networks, 5th Edition**. [S.l.]: Pearson, 2011.

WEBRTC. **iLBC Freeware**. [Online; acessado em 8-Maio-2016], https://webrtc.org/license/ilbc-freeware/.

WEBRTC. **WebRTC**. [Online; acessado em 26-Maio-2016], https://webrtc.org/.

XIPH.ORG. **Speex**. [Online; acessado em 8-Maio-2016], http://www.speex.org/.

XIPH.ORG. **OPUS**. [Online; acessado em 8-Maio-2016], http://www.opus.org/.