Projeto

Mestrado em Gestão de Sistemas de Informação Médica

Servidor de Conferências entre Clientes WebRTC e SIP

Ricardo Martins Domingues

Leiria, Maio de 2018

Projeto

Mestrado em Gestão de Sistemas de Informação Médica

Servidor de Conferências entre Clientes WebRTC e SIP

Ricardo Martins Domingues

Projeto de Mestrado realizado sob a orientação do Doutor Paulo Jorge Gonçalves Loureiro, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, Abril de 2018

Resumo

Este documento pretende apresentar o trabalho desenvolvido e resultados obtidos no desenvolvimento de uma solução de videoconferência. Este projeto visa o desenvolvimento de um servidor de videoconferência capaz de interligar múltiplos *endpoints* que utilizam o protocolo SIP (Session Initiation Protocol) ou a tecnologia WebRTC(Web Real-Time Communication).

Este projeto foi desenvolvido em colaboração com a Collab, uma empresa de software de telecomunicação. O objetivo principal é integrar os produtos da empresa para assim contribuir para uma oferta mais diversificada e abrangente. O âmbito do projeto é o desenvolvimento de funcionalidades de videoconferência para utilizadores já existentes em outros sistemas. Assim, o produto desenvolvido garante a interoperabilidade entre sistemas fazendo uso de múltiplos protocolos de comunicação, bem como desenvolveu novos protocolos e APIs.

A arquitetura definida para este produto também está alinhada com a estratégias comercial da empresa que pretende alargar o seu negócio na *Cloud* comercializado produtos no modelo SaaS (Software-as-a-Service). Este tipo de comercialização trás outros tipos de desafios no desenvolvimento de produtos que visam responder a requisitos como a escalabilidade, a diminuição do tempo de interrupção do serviço, o balanceamento de carga, a segurança, *multitenancy* e a elasticidade para se adaptar a diferentes estratégias de licenciamento e a diferentes casos de utilização.

Palavras-chave: Videoconferência, SIP, WebRTC

Abstract

This document aims to present and explain the results obtained during the development of a videoconference solution. The project comprised the development of a videoconference server capable to interconnect multiple SIP and WebRTC endpoints.

This project was developed in collaboration with Collab, a software company. Its main purpose was to integrate with Collab's products, by improving and making them more appealing to company's customers. The project focus was on the development of videoconferencing capabilities for users that already exist in other systems. In order to make this an easy to integrate product, APIs and new protocols were developed so as to guarantee interoperability among systems.

The product's architecture was developed in line with the company's commercial strategy, which aims to expand its business in the Cloud market, through a SaaS (Software-as- a-Service) model. This approach brings novel challenges when trying to meet requirements such as scalability, zero service down time, load balancing, security, multitenancy and the flexibility to adapt to various licensing strategies and different use cases.

Keywords: Videoconferencing, SIP, WebRTC.

Lista de Figuras

Lista de Siglas

API Application Programming Interface

DDI Direct Dial-In

DTLS Datagram Transport Layer Security

DTMF Dual-Tone Multi-Frequency

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HZ Hertz

ICE Interactive Connectivity Establishment

IP Internet Protocol

ITSP Internet Telephony Service Provider

IVR Interactive Voice Response

JSON JavaScript Object Notation

MCU Multipoint Control Unit

NAT Network Address Translation

PBX Private Branch Exchange

PCM Pulse-Code Modulation

RTCP Real-time Transport Control Protocol

RTP Real-time Transport Protocol

SaaS Software as a service

SDP Session Description Protocol

SGBD Sistema de Gestão de Base de Dados

SIP Session Initiation Protocol

SRTP Secure Real-time Transport Protocol

STUN Session Traversal Utilities for NAT

TCP Transmission Control Protocol

TLS Transport Layer Security

TURN Traversal Using Relays around NAT

UDP User Datagram Protocol

UUID Universally Unique Identifier

VoIP Voice over Internet Protocol

WebRTC Web Real-time Comunication

Índice

1. Introdução

Este projeto foi desenvolvido no âmbito da unidade curricular Dissertação/ Projeto/ Estágio do Mestrado Gestão de Sistemas de Informação Médica da Escola Superior de Gestão e Tecnologia do Instituto Politécnico de Leiria em parceria com a Faculdade de Medicina da Universidade do Porto e tem como principal objetivo o desenvolvimento de um servidor de conferências de áudio e vídeo capaz de interligar múltiplos *endpoints* utilizando o protocolo SIP(Session Initiation Protocol) ou a tecnologia WebRTC(Web Real-time Comunication).

A ideia deste projeto nasce da vontade de adquirir mais conhecimento nas tecnologias utilizadas em *Real-Time Comunication*, em particular na recente tecnologia WebRTC e protocolo SIP, massivamente utilizado em soluções de comunicação, aliada à necessidade da empresa Collab - Soluções Informáticas de Comunicação e Colaboração, S.A. em ter uma solução deste tipo, detetada pelo pedido de múltiplos clientes, para integrar com o produto OneContactPBX por ela comercializado.

Assim, este projeto foi desenvolvido em parceria com a Collab e o âmbito definido pelos requisitos impostos pela empresa e ao qual deu o nome de OneContactMeetingServer.

1.1 Objetivos

Pretende-se com este projeto o desenvolvimento de um servidor de videoconferência, designado de "OneContactMeetingServer". Este servidor deverá incluir a criação de salas de videoconferência, as quais devem poder ser acedidas por aplicações externas que usem a tecnologia WebRTC, o protocolo SIP ou através de uma chamada telefónica para um DD(Direct Dial-In) específico.

Qualquer dispositivo que implemente o protocolo SIP deverá poder ligar-se ao servidor de conferências. No que concerne a tecnologia WebRTC e pelo facto desta não ser um protocolo de sinalização, o que dificulta a interoperabilidade com aplicações externas,

será também desenvolvida uma aplicação Web para testes, utilizando a tecnologia WebRTC, capaz de se ligar a uma sala de conferência implementada neste projeto.

Outro objetivo deste projeto é a integração do OneContactMeetingServer com o OneContactPBX. O OneContactPBX é um produto de comunicação desenvolvido e comercializada pela Collab. A empresa pretende com este novo projeto fornecer aos seus clientes de uma forma integrada uma solução completa de conferências de áudio ou vídeo. Assim, o OneContacMeetingServer deverá ser um sistema aberto, capaz de integrar com outras soluções de comunicação e onde os seus utilizadores não necessitam de estar registados dentro desta solução.

Para o desenvolvimento do OneContactMeetingServer foram definidos os seguintes requisitos:

- Desenvolvimento de um servidor de conferência capaz de suportar *endpoints* SIP e WebRTC simultaneamente numa arquitetura distribuída;
- Criar e gerir salas de conferência;
- Suportar conferências de áudio e vídeo;
- Desenvolver funcionalidades de administração de uma conferência;
- Desenvolvimento de uma API(Application Programming Interface) para administração do servidor;
- Desenvolvimento de uma camada de serviços para gestão de utilizadores de aplicações cliente WebRTC;
- Escalabilidade horizontal (capacidade de adicionar mais máquinas para a distribuição de carga no sistema);
- Redundância;
- Tolerância a falhas;
- Zero down time: Atualizações sem interrupção de serviço

1.2 Enquadramento com OneContactPBX

O OneContactMeetingServer será mais um produto a comercializar pela Collab e estará integrado numa arquitetura de comunicação mais ampla e onde já se encontra o OneContactPBX.

O OneContactPBX é uma solução de comunicação composta por vários componentes, tendo como elemento principal o OnePBX. O OnePBX é um PBX(Private Branch Exchange) virtual capaz de fornecer um serviço semelhante a uma central telefónica. Entre outras funcionalidades, O OnePBX permite criar e gerir todas as extensões que se podem registar através de qualquer aplicação ou telefone que implemente o protocolo SIP.

O OnePBX tem como principal objetivo a sinalização de chamadas. Quando uma extensão se regista no OnePBX fica apta a receber chamadas ou a iniciar chamadas para outras extensões que também estejam registadas.

O OnePBX também permite estabelecer ligações a qualquer ITSP(Internet Telephony Service Provider) através da configuração de SIP Trunks. Desta forma, as extensões registadas no OnePBX estão aptas a receber e a iniciar chamadas para fora do OnePBX, como por exemplo, para a rede móvel.

O OneContactMeetingServer vem satisfazer uma necessidade de negócio da Collab que consiste em ter um produto de comunicação que permita às extensão possuir uma sala de conferências onde outras extensões se podem ligar e comunicar entre si. Também é adicionada a funcionalidade de DDI para que outras pessoas, externas ao OneContactPBX, se possam ligar a uma determinada sala de conferência usando somente um código de autenticação.

Como o OneContactMeetingServer está integrado com a solução do OneContactPBX, a solução desenvolvida não recebe os registos de extensões, pois estas registam-se no OnePBX. Também não valida a origem e permissões da chamada, tomando como pressuposto que o OnePBX validou a extensão de origem. Qualquer pedido de início de chamada para uma sala de conferência, que seja conhecida pelo OneContactMeetingServer, é aceite.

O produto desenvolvido também não disponibiliza nenhuma função de criar ou provisionar as salas de conferência no OneContactMeetingServer. Apenas é disponibilizada

uma API que permite executar estas funções por aplicações externas. Na atual arquitetura estas funções estão implementadas no portal de provisionamento do OneContactPBX.

Como o OneContactMeetingServer possui poucas validações de segurança, devido a não conseguir gerir a informação das extensões que se ligam a uma sala de conferência, os componentes que recebem sinalização SIP devem estar protegidos dentro da rede, onde se encontra o OnePBX, e não devem aceitar mensagens SIP que venham fora da rede.

O OneContactPBX não está apto a gerir *endpoints* WebRTC. Para resolver esta situação foi desenvolvido um novo serviço, o OneWebRTCMeetingRoomService, que funciona como *proxy* para com os clientes WebRTC. Este serviço tem como principais funções a validação da autenticação e a sinalização WebRTC para as salas de conferência. Da mesma forma que as salas de conferência não devem estar expostas a ligações SIP de um componente externo à rede, também não devem estar expostas a ligações WebRTC. Assim, o OneWebRTCMeetingRoomService deve ser instalado na mesma rede que a restante solução.

1.3 Metodologia de Desenvolvimento

Este projeto é essencialmente de carácter prático e foi desenvolvido para responder às necessidades dos clientes da Collab que usam o produto OneContactPBX. O projeto passou por várias fazes de desenvolvimento:

1. A primeira fase do projeto foi o levantamento de requisitos. Este trabalho inclui conversas com alguns clientes interessados na solução. Para além dos requisitos funcionais, foi também feito o levantamento de outros requisitos que se destinavam à melhor preparação do produto para a comercialização no modelos SaaS(Software- as-a-Service). Um grande desafio foi a otimização dos recursos e a gestão do serviço na *Cloud*. Para isso, o produto necessita de funcionar com diferentes clientes em simultâneo, que podem apresentar diferentes licenças e preparado para escalar com a aquisição de novos clientes. Para este tipo de comercialização, também é essencial a redução de tempos de indisponibilidade do serviço, sendo um requisito complicado de satisfazer e que afeta o desenvolvimento e arquitetura de todo o sistema.

2. Após a definição de todos os requisitos, seguiu-se a análise de soluções similares disponíveis no mercado e a comparação com alguns produtos concorrentes. Esta fase ajudou à compreensão do mercado e da oferta, bem como permitiu inteirarnos com as dificuldades subjacentes ao desenvolvimento de uma produto deste tipo.

3. A fase seguinte foi a definição da arquitetura do produto e a sua integração com os produtos já comercializados pela empresa.

4. Por último procedeu-se à implementação e validação do produto.

1.4 Organização do Relatório

Este relatório encontra-se dividido em cinco capítulos que pretendem demonstrar todo o trabalho realizado no desenvolvimento deste projeto.

O capítulo 1 começa por fornecer uma breve apresentação dos objetivos que se pretendem alcançar. De seguida, é feito o enquadramento do projeto no sistema OneContactPBX. Por último é apresentada a metodologia do projeto.

O capítulo 2 têm como objetivo introduzir os protocolos e conceitos utilizados no desenvolvimento de uma solução de vídeo conferência e pelo tipo de *endpoints* suportados neste projeto. Tudo o que é neste capítulo detalhado, é frequentemente referenciado ao longo de todo relatório. Este também pretende introduzir para as várias abordagens utilizadas no desenvolvimento de uma solução de vídeo conferência.

O capítulo 3 descreve de forma detalhada o sistema apresentado, bem como os cenários e casos de uso suportado. Este capítulo tem como principal objetivo a demonstração e justificação das decisões tomadas para fazer face aos requisitos definidos, demonstrar quais os desafios encontrados e os protocolos desenvolvidos para responder às necessidades encontradas.

O capítulo 4 descreve em detalhe todos os componentes desenvolvidos e as principais tarefas de cada um. Pretende-se com este capítulo demonstrar pormenorizadamente o trabalho desenvolvido ao longo deste projeto.

O capítulo 5 pretende demonstrar os resultados obtidos com o desenvolvimento deste projeto. Neste são propostos alguns cenários de teste e divulgados os resultados da sua

execução.

2. Conceitos, Protocolos e Arquiteturas de Comunicação Multimédia

2.1 SIP

O Session Initiator Protocol (SIP) é um protocolo de sinalização, presença e *instant messaging* desenvolvido para iniciar, configurar, modificar e terminar sessões de multimédia; solicitar e entregar presença; e enviar e receber mensagens instantâneas(Johnston, 2009).

2.2 WebRTC

WebRTC é o resultado do esforço da indústria para definir *standards* e para colocar recursos de comunicação em tempo real em todos os *browsers*. O WebRTC está acessível aos programadores através do *standard* HTML5(Hypertext Markup Language 5) e de APIs JavaScript(Johnston and Burnett, 2014).

2.3 SDP

Ao iniciar uma conferência multimédia, seja de VoIP(Voice over Internet Protocol), *streaming* de vídeo ou outras sessões, é necessário transmitir detalhes sobre a descrição das sessão. Por exemplo, informação dos codecs, endereços e outros metadados distribuídos entre os participantes(M. Handley and Perkins, 2006).

O SDP(Session Description Protocol) fornece uma representação padronizada para dos dados da sessão, independentemente de como essa informação é transportada. O SDP é apenas um formato para a descrição da sessão pelo facto desta informação não estar

incorporada no protocolo de transporte. O SDP destina-se a ser usado por diferentes protocolos de comunicação conforme apropriado(M. Handley and Perkins, 2006).

Na Figura 3 está apresentado um SDP de exemplo.

```
v=0
o=- 3727877181 3727877181 IN IP4 172.18.190.117
s=OneMeetingRoom
b=AS:1029119
t=0 0
m=audio 8000 RTP/AVP 97 8 0 3 104 98 99 9 96
c=IN IP4 172.18.190.117
b=TIAS:64000
a=rtcp:8001 IN IP4 172.18.190.117
a=sendrecv
a=rtpmap:97 speex/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:3 GSM/8000
a=rtpmap:104 iLBC/8000
a=fmtp:104 mode=30
a=rtpmap:98 speex/16000
a=rtpmap:99 speex/32000
a=rtpmap:9 G722/8000
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-16
```

Figura 3: Exemplo de SDP

2.4 RTP

O RTP (Real-time Transport Protocol) é um protocolo de transporte desenvolvido para aplicações que necessitam de trocar dados em tempo real, como por exemplo, dados de áudio ou vídeo. Este protocolo é complementado por um protocolo de controlo, RTCP (Real-time Transport Control Protocol), o qual permite a monitorização da entrega dos dados nos pacotes RTP e fornece funcionalidades de controlo e de identificação. Estes dois protocolos são independentes das camadas de transporte e de rede(H. Schulzrinne and Jacobson, 2003).

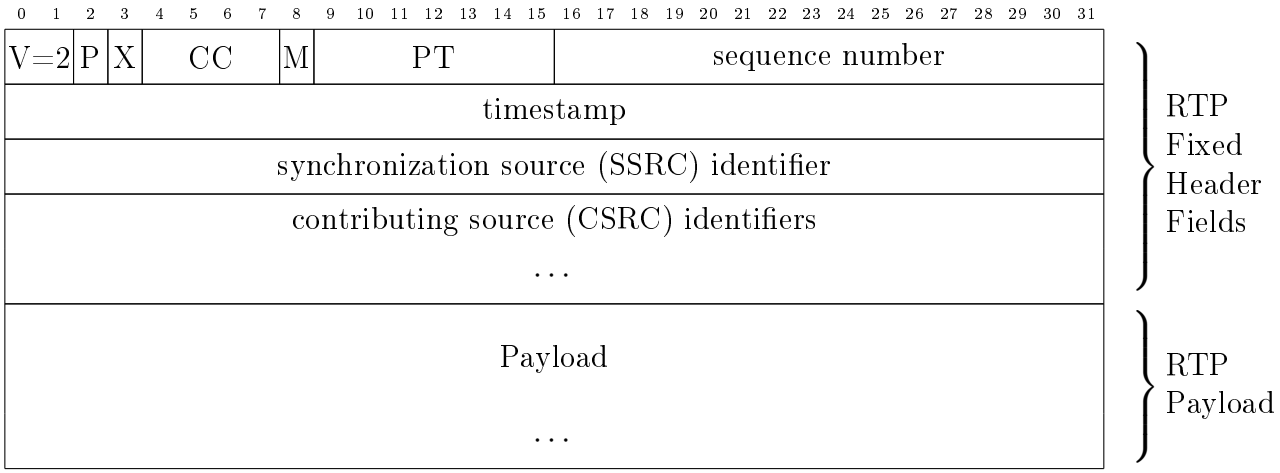

Figura 4: Formato RTP

Na Figura 4 é apresentado o formato do protocolo RTP. Os campos apresentados têm o seguinte significado:

- **Version (V)**. Este campo tem o número da versão do RTP, devendo ser sempre o valor 2 para estar de acordo com o RFC-3550.
- **Padding (P)**. Este bit identifica a existência de um ou mais octetos de preenchimento que não fazem parte do *payload*. Se o valor do bit for 1, deve ser lido o último octeto que indicará o número de octetos a serem ignorados, incluído ele próprio. O preenchimento de dados que não fazem parte do *payload* pode ser necessário para alguns algoritmos de encriptação que usam blocos de tamanho fixo.
- **Extension (X)**. Este bit marca se existe um *header* de extensão. O RTP pode ter mais um *header* de extensão que não está representado na Figura 4.
- **CSRC count (CC)**. Este é o número de identificadores CSRC contidos no pacote RTP.
- **Marker (M)**. É um bit que marca eventos significantes. Um exemplo de uso deste marcador é nas *streams* de vídeo para marcar que os dados enviados no *payload* correspondem ao início de uma nova frame.
- **Payload Type (PT)**. Este campo identifica o formato do *payload* e determina a sua interpretação pela aplicação. Nas *streams* de áudio ou vídeo é preenchido com o identificador do codec.
- **Sequence number**. É um valor que é incrementado a cada pacote enviado e é usado para deteção de pacotes perdidos.

- **Timestamp**. *Timestamp* que reflete o instante em que a primeira sample do *payload* foi retirada.

- **SSRC**. O SSRC identifica a fonte de sincronização. Um exemplo da utilização deste campo é em *streams* que usam a técnica de *multiplexing*, que é usada neste projeto pelos participantes WebRTC, em que várias *streams* de áudio e vídeo são transformadas em uma *stream* e é usado este valor para identificar qual a *stream* que originou o pacote recebido.

- **CSRC list** Lista de identificadores CSRC. A finalidade desta lista não é aqui explicada por não ser utilizada pelos componentes desenvolvidos neste projeto.

- **Payload** No caso de uma *stream* de *áudio* ou *vídeo* é aqui que está a informação gerada pelos codecs.

2.5 Sinalização

Os processos de inicialização e conclusão de chamadas envolve a sinalização do subscritor com o seu *exchange*(Bosse and Devetak, 2007). No caso do produto OneMeetingServer o *exchange* é o OnePBX e os subscritores são todas as extensões nele registadas. Ao iniciar uma chamada para uma sala de conferência, uma extensão necessita de iniciar um processo de sinalização para se juntar à respetiva sala através de um protocolo definido. Na solução OneContactPBX os subscritores podem ser de vários tipos: *Hardphones*, *Softphones* e aplicações específicas disponibilizadas pelo produto. Todas estas interfaces utilizam o protocolo SIP para comunicar com o OnePBX. SIP é o único protocolo de sinalização que o OnePBX implementa para gestão de chamadas. Por esta razão é fundamental que o OneContactMeetingServer disponibilize uma interface SIP para melhor integrar com o OnePBX. Outro objetivo do OneContactMeetingServer é fornecer uma interface Web desenhada especificamente para interagir com uma sala de conferências. Deste modo, o OneContactMeetingServer implementa o *standard* WebRTC para facilitar a integração com os *browsers* atuais.

O processo de sinalização não é padronizado no WebRTC, apenas é considerado parte da aplicação. A sinalização pode ser executada sobre protocolo HTTP(Hypertext Transfer Protocol) ou pelo uso de WebSockets. O servidor de sinalização poderá estar num servidor Web que também suporte a sinalização ou num servidor específico que apenas lide com a sinalização(Johnston and Burnett, 2014). Para o OneContactMeetingServer foi desenvolvido um protocolo de sinalização específico que suporta a

solução das salas de conferências. Este protocolo foi desenvolvido sobre o *standard* JSON(JavaScript Object Notation) para facilitar a interação com os *browsers*. Também foi desenvolvido o componente OneMeetingWebRTCService com o objetivo de suportar a sinalização de *endpoints* WebRTC por via de WebSockets.

2.6 Audio/Video Coding

Audio Coding é uma técnica que permite obter uma representação compacta de um sinal de áudio ou vídeo digital com o objetivo de tornar a sua transmissão ou armazenamento mais eficiente (Painter and Spanias, 2000).

A técnica consiste em representar o sinal no número mínimo de *bytes* possível sem perturbações significativas no sinal original e com consumo reduzido dos recursos de processamento (Pan, 1993)(Painter and Spanias, 2000).

O *Audio Coding* é essencial na diminuição do consumo de largura de banda em comunicação de áudio e vídeo em tempo real. Esta tarefa é desempenhada por múltiplos codecs que são negociados entre o transmissor e o recetor.

2.7 Amostragem de áudio

Amostragem é o processo de transformação de um sinal analógico para um sinal digital. Deste processo resultam amostras (*samples*) que são retiradas do sinal analógico num determinado intervalo de tempo. Quanto maior a frequência de amostragem (*sample rate*) melhor é a aproximação do sinal digital ao sinal analógico. Também o tamanho definido para a amostra influencia a qualidade da representação. Por exemplo, uma amostra de 16 bits comporta uma maior gama de valores do sinal do que uma amostra com 8 bits(Engineering, 1997)(Lavry, 2004).

A amostragem de sinais de áudio é o processo de transformação especifico de um sinal analógico para um sinal de áudio digital. A taxa de amostragem é medida em Hertz (Hz). Por exemplo, um CD de áudio usa uma taxa de amostragem de 44100Hz e tem uma precisão, ou tamanho da mostragem, de 16 bits permitindo a que cada amostra represente 65536 níveis diferentes(Lavry, 2004).

2.8 Audio Resampling

Para que seja possível executar operações sobre os sinais de áudio, como a mistura de áudio, é necessário que todos se encontrem na mesma frequência. O *Audio Reampling* é uma técnica que visa a conversão da frequência da amostragem de um sinal de áudio digital numa outra frequência (Hentschel and Fettweis, 2001).

Cada codec define a sua frequência de operação. Assim, é necessário o uso desta técnica para que o áudio de todos os participantes do OneContactMeetingServer possa ser misturado e distribuído na frequência correta para cada participante.

2.9 Videoconferência

Videoconferência é um termo utilizado para descrever um sistema capaz de realizar uma reunião à distância entre dois ou mais participantes e cada participante pode ver, ouvir e falar com os outros em tempo real e, em sistemas mais avançados, partilhar conteúdo entre dispositivos durante a conferência.(Marquardt, 2001)(Sheet and Sheet, 2008). Os componentes básicos de *hardware* necessários à realização de uma videoconferência são: o microfone, a câmara, as colunas e ecrã para mostrar o vídeo.

Para a realização de uma videoconferência são necessários os seguintes *softwares*:

- *Software* cliente para capturar a voz e a imagem, fazer a codificação e para transmitir o sinal destinado aos outros participantes. Simultaneamente, no recetor, fazer o processo inverso, ou seja, receber o sinal digital de voz e vídeo, fazer a descodificação e reproduzir o áudio e o vídeo(Sheet and Sheet, 2008).

- *Software* que interliga todos os participantes e que gere a troca do sinal de voz e vídeo entre eles. Em qualquer *endpoint*, o sinal de voz e vídeo é combinado e entregue aos restantes *endpoints* em forma de fluxo em tempo real(Sheet and Sheet, 2008).

- Opcionalmente, pode existir um ferramenta de gestão capaz de agendar sessões de videoconferência(Sheet and Sheet, 2008).

2.10 Videoconferência ponto a ponto

Videoconferência ponto a ponto é um tipo de videoconferência limitado a duas localizações, ou seja, está limitado a dois participantes.

2.11 Videoconferência multiponto

As conferências multiponto estão estruturadas em dois modelos: o centralizado e o descentralizado. No modelo centralizado cada participante envia e recebe a sinalização de um MCU (*Multipoit Control Unit*) ou *media bridge*(Weinstein, 2012). O MCU é o dispositivo responsável por receber as *streams* de todos os participantes, juntar o sinal de áudio e de vídeo e reenviar o sinal para todos os participantes. O modelo MCU está representado abaixo na Figura 6.

No modelo descentralizado cada participante negoceia o vídeo e áudio diretamente com os restantes participante na videoconferência. Este modelo não requer um equipamento MCU mas torna o processo de gestão mais complexo e aumenta a utilização da largura de banda e do processamento com o aumento do número de participantes(Weinstein, 2012). Este está representado na Figura 5.

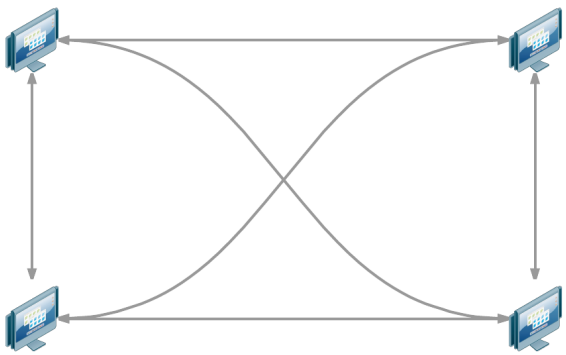

Figura 5: Descentralized Multipoint

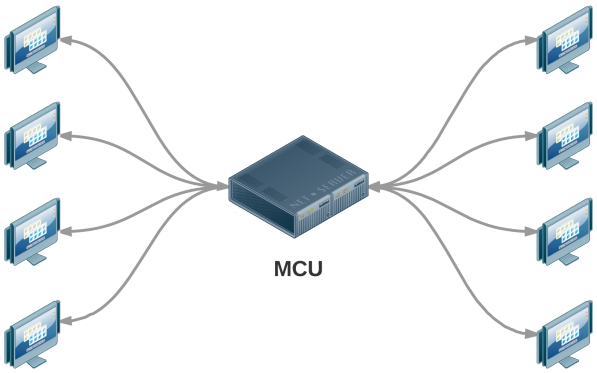

Figura 6: MCU

2.12 Arquiteturas para servidores de conferências

As arquiteturas para um servidor de conferências podem ser classificadas em dois modelos: o centralizado e o distribuído. A arquitetura centralizada fornece múltiplos serviços para os *endpoints* da conferência mas necessita de um equipamento específico para cada serviço. Esta abordagem é a mais comum para os sistemas de conferência de áudio e vídeo. A arquitetura centralizada fornece um ponto único para administração e gestão do serviço. Para adicionar uma nova funcionalidade é apenas necessário a atualização de um componente do sistema(Firestone et al., 2007).

Para suportar um grande número de participantes, o sistema de conferências é estruturado em vários componentes diferentes. Cada um destes componentes, separados em diferentes máquinas e espalhados por várias localizações na rede, comunicam entre si para formar um sistema único(Firestone et al., 2007). Embora a arquitetura distribuída seja mais complexa de desenvolver, manter, atualizar e configurar, sem ela estaríamos limitados à capacidade de processamento, armazenamento e largura de banda de uma única máquina e à dificuldade em garantir a disponibilidade do serviço no momento em que surge um problema ou quando é feita a atualização do sistema.

2.13 Produtos de Videoconferência

No desenvolvimento deste projeto foram investigados e analisados produtos para vídeo conferências capazes de suportar participantes SIP e WebRTC. Entre algumas soluções analisadas, as que cumpriram os requisitos principais foram as soluções WebRTC2SIP,

Licode e OpenVCX. Destas três soluções, a única que se conseguiu instalar com êxito foi o OpenVCX mas os testes falharam quando se tentou utilizar a aplicação WEB que esta disponibilizava. Quando se tentou iniciar a aplicação no Google Chrome e no Mozilla Firefox, esta apresentou múltiplos erros e dificuldades na negociação dos canais de media. Alguns erros na aplicação de teste ainda se conseguiram resolver mas nunca se foi capaz de realizar uma conferência de dois participantes.

Também nenhuma das soluções apresentadas estudadas pareceu de fácil integração com o OneContactPBX e não apresentavam mecanismos para escalabilidade, sendo necessário desenvolvimento de múltiplos produtos complementares para conseguir satisfazer os requisitos.

2.14 Síntese

Este capítulo apresentou os vários protocolos utilizados no desenvolvimento deste projeto na sinalização e comunicação dos diferentes *endpoints* com a solução desenvolvida. Também foram apresentados os principais conceitos inerentes a uma solução de videoconferência, como a codificação, amostragem de áudio, reamostragem, entre outros. Por fim é feita a descrição das várias arquiteturas possíveis para o desenvolvimento deste tipo de produto. O capítulo seguinte apresenta a solução desenvolvida, o OneContactMeetingServer, de forma geral, apresentando ainda os principais desafios, a arquitetura escolhida, os cenários suportados e a justificação para algumas das decisões tomadas.

3. Produto Desenvolvido OneContact-MeetingServer

O OneContactMeetingServer é o servidor desenvolvido para suportar videoconferências. Este capítulo pretende introduzir o desenvolvimento do OneContactMeetingServer, dando a conhecer as principais dificuldades no desenvolvimento de um servidor de videoconferência, bem como algumas dificuldades adicionais devido à arquitetura escolhida. Também são apresentados os protocolos desenvolvidos para dar resposta aos problemas encontrados.

Este capítulo também aborda os principais componentes que compõem este produto, os tipos de salas de conferências que podem ser criadas no servidor, que casos de utilização estas servem e apresenta a justificação das decisões tomadas durante o desenvolvimento e definição da arquitetura.

3.1 Desafios

Após a ideia de desenvolver um servidor de videoconferências, capaz de suportar *endpoints* SIP e WebRTC, foram de imediato identificados vários desafios. Estes dois tipos de *endpoints* não são compatíveis entre si e envolvem o conhecimento de múltiplos protocolos que são extensos e complexos. Desde o início que havia o conhecimento de que a execução de uma sala de conferência consome bastantes recursos da máquina que a suporta, nomeadamente recurso de CPU e largura de banda. O final do processo de levantamento de requisitos trouxe também novos desafio que tiveram de ser considerados, principalmente com a definição de requisitos não funcionais, como a escalabilidade e a alta disponibilidade com o objetivo de atingir o *zero down time* do serviço. Com o decorrer do desenvolvimento do projeto foram surgindo mais desafios, principalmente devido à complexidade do sistema idealizado para conseguir dar resposta a todos os seus requisitos, funcionais e não funcionais inicialmente definidos mas também por pequenos pormenores que só se conseguem ter em consideração depois de

alguma experiência no desenvolvimento deste tipo de soluções.

3.1.1 Escalabilidade horizontal

A escalabilidade horizontal é um dos requisitos não funcionais definidos e é um dos requisitos mais desafiantes. Este e o requisito de *zero down time* do serviço são aqueles que mais influência tiveram na definição da arquitetura deste sistema. A arquitetura foi desenhada de forma a permitir, em qualquer momento, adicionar uma ou mais máquinas para suportar a carga exercida pela execução de mais salas de conferência. O sistema conta ainda com um componente central, o OneMeetingServerManager, que faz a gestão de todas as máquinas que estão disponíveis e que define em que nó é que uma sala é executada. Para que o OneMeetingServerManager não se torne um *bottleneck* do serviço, o sistema foi arquitetado em componentes facilmente balanceados, tendo em vista o isolamento de todas as tarefas que exigem um maior esforço, tanto no CPU com na rede e que necessitam de consultar o OneMeetingServerManager com baixa frequência, ficando assim exclusivamente alocado a poucas tarefa e que exigem baixo esforço.

3.1.2 Performance

Implementar uma sala de conferências implica o desenvolvimento de tarefas que requerem o uso intensivo de processador, tais como, processar os pacotes de *media* com elevada frequência, tipicamente de 20 em 20 milissegundos para cada *media* de cada participante, codificar e descodificar os medias, misturar os sinais de áudios de todos os participantes, efetuar *resampling*, entre outras. Para alcançar o melhor desempenho é necessário a adoção de tecnologias de baixo nível, que não façam uso de *frameworks* pesadas e que ofereçam maior liberdade na gestão de memória, mesmo que essas tecnologias impliquem maior complexidade e tempo de desenvolvimento. Também é necessário escolher uma arquitetura inteligente do *software*, com o intuito de reduzir ao máximo o número de instruções e de cálculos, principalmente nas tarefas de elevada frequência. Relativamente ao processamento dos *medias*, é necessário fazer a escolha correta de *codecs* e da frequência ideal para a amostragem processadas pelo misturador que causem menos impacto na utilização dos recursos. Assim será necessário escolher frequências de amostragem mais baixas, evitar o uso da técnica de *resampling* e ao mesmo tempo escolher os *codecs* com melhor compromisso entre uso do CPU e largura de banda necessária.

3.1.3 *Zero down time*

Como já foi referido na secção da escalabilidade, *Zero down time* do serviço é um dos requisitos que mais influenciou o desenho da arquitetura do sistema. Também como referido que, grande parte dos componentes, foram desenhados para serem balanceados. Esta característica possibilita a disponibilidade do sistema mesmo que uma máquina ou o componente esteja temporariamente indisponível. No caso do OneMeetingServerManager, o único componente central do sistema e que não é balanceado, foi acrescentado um mecanismo de redundância que trabalhará em paralelo e em sincronia com outra instância localizada numa outra máquina.

Somente os mecanismos de balanceamento e redundância não são suficientes para garantir o objetivo de *zero down time* do serviço. No cenário de uma atualização ao sistema, é necessário garantir que este consegue retirar uma máquina, deixando terminar primeiro todas as conferências em execução, e que todos os componentes continuem a comunicar mesmo que com outros em versões diferentes. Só assim é conseguida a atualização gradual do sistema sem que este se torne indisponível. Para que todos os componentes comuniquem entre si em versões diferentes é ainda necessária a adoção de mecanismos de troca de mensagem com estrutura flexível, que não definam campos obrigatórios e consigam manipular mensagens com campos em falta ou campos a mais do que são esperados numa determinada versão. Em relação a persistência dos dados é também necessário um sistema flexível, que permita leituras e escritas dos documentos em várias versões e que se distancie da lógica das base de dados relacionais convencionais.

3.1.4 Rede

Com o decorrer do desenvolvimento do OneContactMeetingServer foram aparecendo alguns desafios relacionados com a comunicação em rede, nomeadamente com os canais de *media* entre os clientes e a sala de conferência.

O primeiro desafio tinha a ver com o facto dos clientes, maioritariamente, estarem situados atrás de *routers* com *firewalls* e mecanismos de NAT(Network Address Translation). Esta configuração de rede dificultava a correta negociação dos canais impossibilitando a receção dos pacotes de áudio e vídeo provenientes da sala de conferência. Embora existam protocolos que resolvem este problema, como é o caso do protocolo ICE(Interactive Connectivity Establishment), este implicava algumas configurações complexas na parte

do cliente, principalmente nos clientes SIP, o que o afastava de uma solução viável.

A gestão dos portos disponíveis para os canais de vídeo foi outro desafio detetado após algum estudo sobre a negociação dos mesmos. Por exemplo, na solução mais lógica para N participantes com vídeo ativo seriam necessários $2 * (N + 1) * N$ portos. Se fosse utilizada esta abordagem, rapidamente se esgotaria o número de portos de *media* disponíveis.

3.2 Arquitetura do OneContactMeetingServer

A solução OneContactMeetingServer é um MCU que implementa uma estrutura, na qual todos os participantes estão ligados a um componente que recebe e fornece os sinais de áudio e o vídeo de todos os participantes e está desenvolvido seguindo uma arquitetura distribuída para responder a requisitos como a estabilidade e a alta disponibilidade com *Zero down time* do serviço.

O OneContactMeetingServer está estruturado em 3 componentes principais, o OneMeetingServerManager, o OneMeetingPool e o OneMeetingRoom. Cada componente representa um processo diferente que será detalhado posteriormente neste documento. A decisão de separar o servidor em três componentes tem como objetivo principal responder aos requisitos de escalabilidade horizontal, redundância e atualizações sem tempo de interrupção de serviço.

Numa instalação da solução OneContactMeetingServer deverá haver duas instâncias do componente OneMeetingServerManager, instaladas em máquinas diferentes. O objetivo destas duas instâncias é garantir redundância ao serviço. Quando a instância utilizada falha, a outra deve ser capaz de carregar o estado da que falhou e continuar a suportar o serviço.

O OneMeetingServerManager é o gestor de toda a solução, sendo o único que tem uma base de dados onde armazena todos os dados relativos ao provisionamento, ao *runtime* e ao histórico. Desta forma, este apresenta três APIs principais, uma para provisionamento, outra para pesquisar ou agir sobre salas que estão em execução e por último um API que permite a consulta de dados de histórico e de *reports*.

O OneMeetingPool é um serviço em execução numa máquina e faz a gestão das OneMeetingRooms dessa máquina. Só pode haver uma instalação deste componente por cada

máquina. Em cada máquina onde estiver instalado o OneMeetingPool, é configurado o endereço IP(Internet Protocol) do OneMeetingServerManager que deve ter as duas instâncias disponíveis através de um mecanismo de *failover cluster*. Cada vez que uma instância deste componente inicia, deve-se registar no OneMeetingServerManager para que este possa adicionar a OneMeetingPool ao conjunto de *pools* disponíveis para esta solução. A escolha de uma arquitetura por *pools* distribuída em várias máquinas tem como objetivo responder ao requisito de escalabilidade horizontal.

Os processos que suportam a execução das OneMeetingRooms são aqueles que consomem mais recursos de CPU, memória e tráfego de rede. Por esta razão é fundamental que estes processos possam ser distribuídos por várias máquinas. É da responsabilidade do OneMeetingPool orquestrar todas as OneMeetingRooms de uma determinada máquina.

Também é nas OneMeetingPools que está configurada a coleção de portos que é utilizada pelos vários processos que suportam cada uma das OneMeetingRoom. Por somente um processo poder estar à escuta em um determinado porto, será configurado nas OneMeetingPools uma gama de portos para a sinalização SIP e WebRTC e ainda uma gama de portos de *media*, essenciais para as OneMeetingRooms receberem e enviarem pacotes de áudio e vídeo.

A OneMeetingPool tem a responsabilidade de iniciar os processos das OneMeetingRooms e de fornecer os portos UDP(User Datagram Protocol) e TCP(Transmission Control Protocol) para a sinalização SIP e os portos TCP para a sinalização WebRTC. Deverá ainda ser atribuída uma gama portos de *media*, logo no início do processo. A necessidade de portos de *media* crescerá à medida que se juntarem mais clientes à conferência e, por isso, a OneMeetingPool terá que alocar portos suficientes para garantir a entrada de participantes na sala até ao seu limite máximo.

Outra função do OneMeetingPool é fazer o cálculo do nível de carga com base na utilização do CPU, da memória, da rede, dos portos configurados ainda disponíveis e no número OneMeetingRooms ativas na máquina. Periodicamente, a OneMeetingPool, comunica o seu nível de carga ao OneMeetingServerManager, pois é este o componente que faz o balanceamento de carga entre os vários *pools* com base na carga das máquinas (existe ainda outros algoritmos possíveis para distribuição de carga nas máquinas como por exemplo Round Robin).

Com esta arquitetura, o OneMeetingServerManager é o componente que tem o conhecimento completo do estado da solução. É a ele que são feitos os pedidos de criação de

uma nova sala com um determinado identificador, que valida os componentes de segurança e valida as permissões e, em seguida, pede a uma OneMeetingPool (idealmente aquela que tiver menos carga) para iniciar uma sala de conferência.

Após o pedido de início, a OneMeetingPool responde ao OneMeetingServerManager com as informações necessárias para aceder à sala iniciada. Esta informação inclui o endereço IP, os portos UDP e TCP para os *endpoints* SIP e os portos TCP para a sinalização do WebRTC.

3.3 Comunicação e Sinalização

No processo de desenvolvimento da solução OneContactMeetingServer foi necessário o uso de diversos protocolos já existentes e o desenvolvimento de novos protocolos para endereçar problemas específicos e que vão ser explicados nesta secção.

Comunicação entre OneMeetingServerManager e OneMeetingPool

O OneMeetingServerManager é responsável por eleger a OneMeetingPool onde uma sala de conferências decorre. É imperativamente necessária a comunicação entre estes dois processos, que poderão estar em máquinas distintas, para que esta negociação seja possível.

A comunicação entre estes dois componentes é estabelecida através da negociação de um *Socket* TCP e a serialização/desserialização de mensagens usando o mecanismo Protocol Buffers.

O OneMeetingServerManager é responsável por ficar à escuta de novas conexões TCP num porto previamente configurado. As OneMeetingPools é que tomam a iniciativa de negociar uma conexão TCP para uma combinação IP/Porto configurada.

Posteriormente à negociação TCP, poderão ser trocadas mensagens nos seguintes contextos:

- **Registo da OneMeetingPool**. Primeira sequência de mensagens a ser trocada. Permite que a OneMeetingPool, que enviou o registo, fique elegível para receber

salas de conferências.

- **Criação de Sala**. Sequência de mensagens iniciada pelo OneMeetingServerManager com o objetivo da OneMeetingPool criar uma nova OneMeetingRoom e onde fica à espera de resposta com a informação dos *endpoints* SIP e WebRTC da nova OneMeetingRoom.
- **Reportar carga da Pool**. Mensagem enviada por OneMeetingPool com um indicador de carga que o OneMeetingServerManager utiliza para determinar em que OneMeetingPool vai ser iniciada um nova sala.

Comunicação entre OneMeetingPool e OneMeetingRoom

As OneMeetingRooms são processos geridos pelas OneMeetingPools, sendo que cada OneMeetingPool controla unicamente as OneMeetingRooms da sua máquina. Devido ao facto de os processos que necessitam de comunicar entre si se encontrarem em execução na mesma máquina,a estratégia de comunicação escolhida para estes dois processo foi com recurso a *NamedPipes* e serialização/desserialização usando o Protocol Buffers.

A implementação de *NamedPipes* no sistema operativo Windows disponibiliza uma funcionalidade que facilita a serialização e desserialização das mensagens comparativamente com a utilização de *Sockets* TCP. Enquanto os *Sockets* TCP são exclusivamente *stream oriented*, os NamedPipes, no Windows, são possíveis de configurar para serem *stream oriented* ou *message oriented*.

A comunicação com o paradigma *message oriented* permite a uma aplicação ler um determinado conjunto de bytes e perguntar ao sistema operativo se aquela mensagem já foi lida na totalidade ou é necessário fazer um pedido de leitura de mais bytes. Na comunicação *stream oriented* não é possível determinar se um determinado número de bytes lido representa uma mensagem, parte de uma mensagem ou até múltiplas mensagens. Os *NamedPipes*, ao contrário dos *Sockets* TCP, não obrigam à implementação suplementar de um protocolo, que irá ser apresentado mais à frente neste documento, que soluciona este problema.

Os *NamedPipes* são negociados com base em um nome que os dois processos conhecem. Depois, o processo servidor fica à escuta que o processo cliente se conecte. Neste caso o processo servidor é a OneMeetingPool que gera um nome com um UUID(Universally Unique Identifier) o qual é passado por argumento quando a OneMeetingPool inicia.

Neste componente são trocadas as seguintes mensagens:

- **Sala preparada**. Quando a sala inicia, esta necessita de executar um conjunto de tarefas que podem demorar alguns segundos; quando a sala está pronta a receber participantes, esta envia uma mensagem para OneMeetingPool a informar que se encontra preparada.

- **Sala escolhida**. As OneMeetingPool não iniciam OneMeetingRooms somente quando necessitam. Como as OneMeetingRooms podem demorar alguns segundos a ficarem prontas, as OneMeetingPools tem um conjunto de salas, tipicamente duas, prontas a funcionar quando necessário. Quando iniciada, a OneMeetingRoom envia uma mensagem a informar que se encontra preparada e fica a aguardar a mensagem de sala escolhida. Esta mensagem para além de informar uma OneMeetingRoom que foi escolhida, também contém informações sobre a conferência que vai suportar.

- **Sala em execução**. Esta mensagem é a que sucede à mensagem de sala escolhida. É enviada pela OneMeetingRoom para informar que processou a informação recebida e que já se encontra ativa para receber conexões dos participantes.

3.3.1 OneMeeting Message Protocol

Como foi referido anteriormente, foi necessário o desenvolvimento de um protocolo para que a comunicação *stream oriented* dos *Sockets* TCP fosse compatível com as mensagens com um início e um fim definido. Este protocolo é relevante para o bom funcionamento dos mecanismos de serialização/desserialização.

O OneMeeting Message Protocol consiste no envio de 4 bytes iniciais que constituem o *Header* mais o *Payload* que contém a mensagem a ser desserializada, representado na Figura 7 . Os primeiros 2 bytes do *Header* correspondem a um número de sequência e os outros 2 ao tamanho do *Payload*

O parâmetro sequência é um valor numérico que é iniciado a zero e incrementado uma vez a cada mensagem enviada até ao valor de 65534 ($2^{16} - 2$). Este parâmetro tem como objetivo validar a integridade da comunicação. As duas partes da conexão têm o seu próprio valor de sequência e devem verificar se o valor recebido é o esperado. Quando o valor de sequência é corrompido, a conexão deve ser encerrada e tratada

como um erro. Uma vez atingido o valor máximo de sequência, este deve ser reiniciado com o valor zero.

O tamanho da mensagem tem como objetivo preparar a leitura da mensagem. Depois de recebido um determinado número de bytes, este deve ser usado para perceber até onde uma mensagem deve ser lida ou se é necessário aguardar pela chegada de mais alguns bytes dessa mensagem. Quando o número de bytes recebidos é maior que o valor que está no tamanho da mensagem, os bytes excedente são tratados como o início de uma nova mensagem na sequência.

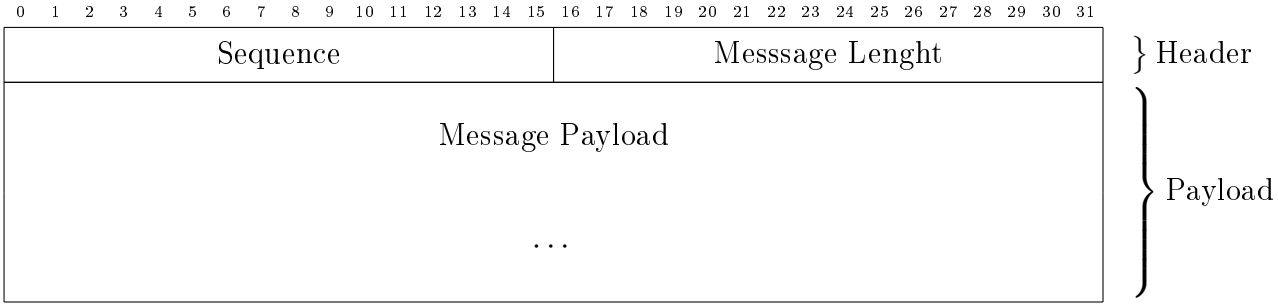

Figura 7: OneMeeting Message Protocol

O OneMeeting Message Protocol também disponibiliza um mecanismo de envio de mensagem de *Keep-Alive*. Estas mensagens destinam-se à manutenção da conexão em longos períodos evitando que componentes na rede fechem as conexões por inatividade.

A mensagem de *Keep-Alive* é um pacote constituído pelos 4 bytes do *Header* e que tem uma sequência de bits que nunca é utilizada no envio das outras mensagens. Nesta mensagem todos os bits correspondentes à sequência têm o valor de 1, o que dá um valor de sequência de 65535. O valoe máximo possível de ser utilizadp nas mensagens é 65534. O valor dos restantes bits do *Header* é alternado entre 1 e 0. A mensagem de *Keep-Alive* está representada na Figura 8

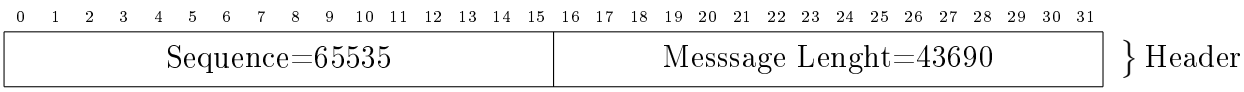

Figura 8: OMMP - Keep-Alive

3.3.2 Sinalização WebRTC

Ao contrário do protocolo SIP, o WebRTC não define um protocolo de sinalização, sendo necessário o desenvolvimento de um que permita os clientes sinalizarem as suas

salas. O protocolo desenvolvido foi adaptado do JsonRTC Protocol(Oracle, Oracle) e está representado na Figura 9.

```
{
  "control": {
    "type" : "request|response|message|acknowledge|error",
    "subsession_id": "<ID da sub-sessao ou trasacao>",
    "sequence" : <numero de sequencia>,
    "message_state" : "Initial|Subsequent|Final",
    "version" : "<versao do protocolo>"
  },
  "header" : {
    "action" : "start|complete|info|notify|shutdown",
    "target" : "<destinatario>",
    "initiator" : "<remetente>",
    "name" : "<nome do remetente>",
    "error" : <codigo de erro>,
    "auth" : "<token de autenticao>"
  },
  "payload" : {
    "sdp" : "<SDP>",
    "type" : "offer|answer",
    "ids" : [
      "<id1>",
      ...
      "<idN>"
    ],
    "id" : "<id do participante da mensagem>"
    "info" : [
      {
        "id" : <id1>,
        "name" : "<nome do participante com o id1>",
        "video" : "true|false"
      },
      ...
      {
        "id" : <idN>,
        "name" : "<nome do participante com o idN>",
        "video" : "true|false"
      }
    ]
  }
}
```

Figura 9: Protocolo Sinalização WebRTC

Como pode ser verificado na Figura 9, o protocolo é constituído pelos seguintes com-

ponentes principais:

- **Control Header**. Este é o primeiro campo que a aplicação deve ler para tratar uma determinada mensagem. Este contém o tipo da mensage; o ID da sessão (ID que identifica uma interação); o ID da sub-sessão ou transação (ID que identifica uma sub-sessão ou transação dentro de uma sessão); o número de sequência da mensagem (número inteiro incrementado cada vez que uma mensagem é enviada); o estado da mensagem (se não for enviado é assumido o valor "Intial"); e a versão do protocolo.

- **Generic and Action Header**. Este contém informação especifica de uma interação. Este componente tem como propriedades o destinatário (a quem se destina a interação); o remetente (quem iniciou a interação); o nome do remetente (nome a ser mostrado); a ação (propósito da mensagem); o código de erro (no caso de uma mensagem de erro); e a autenticação (*token* gerado no processo de autenticação ou vazio se não autenticado).

- **Message Payload**. Este é o componente que define o corpo da mensagem e pode conter o SDP; o tipo de SDP; ids (lista de todos os ids de todos os participantes em uma conferência); id (id do participante que está a receber a mensagem); e a informação dos participantes (descreve os atributos de cada participante para que uma aplicação possa apresentá-los).

3.4 Tipos de salas

Existem dois tipos principais de sala de conferência, as *Instant Meeting Rooms* e as *Owned Meeting Rooms*. As *Instant Meeting Rooms* são salas que não têm um dono e podem ser iniciadas a qualquer altura no sistema sem se encontrarem previamente provisionadas e por qualquer utilizador. As *Owned Meeting Rooms* são salas que são provisionadas previamente e só o seu dono é que as pode iniciar.

Em seguida são apresentadas as necessidades endereçadas por cada um dos dois tipos de sala, sendo que a disponibilidade de cada um dos cenários depende da configuração do OneContactPBX. É assim possível a disponibilização de cada um dos cenários em separado dependendo do que o cliente pretende ou tem contratualizado.

Instant Meeting Rooms

As *Instant Meeting Rooms* respondem a três tipos de necessidades diferentes que são frequentemente apresentadas pelos clientes da Collab que utilizam o OneContactPBX.

A primeira é na utilização das várias aplicações de comunicação disponibilizadas pelo produto. Estas aplicações são geralmente usadas como ferramenta de trabalho, permitem fazer chamadas por SIP, enviar mensagens, gerir contactos e publicar estados (online, offline, Busy, etc) e são disponibilizadas em diferentes ambientes (mobile, web e desktop). É frequentemente solicitado o desenvolvimento de funcionalidades que permitam a estas aplicações adicionar novo participantes no decorrer de uma chamada.

A solução encontrada até ao momento foi suportar a conferência no dispositivo de um dos participantes. Esta solução é bastante limitada para aquilo que são as expetativas dos clientes, pois todo o processamento da conferência é feito localmente por uma das aplicações de cliente e esse nunca pode abandonar a conferência, uma vez que é ele o responsável por unir todos participantes. Estas aplicações também têm grandes limitações quanto ao número de participantes uma vez que as máquinas que as suportam podem ter recursos muito limitados.

As *Instant Meeting Rooms* vêm resolver o problema apresentado. Quando existe uma chamada a decorrer entre duas pessoas e se adiciona uma terceira pessoa, o OnePBX faz um pedido através do protocolo SIP para o OneContactMeetingServer, para que este possa iniciar uma sala e em seguida possa transferir as três chamadas para a nova sala. Quando alguém pretende adicionar um novo participante à conferência, a aplicação irá realizar uma chamada em paralelo para o novo participante e irá transferi-lo automaticamente para a sala, logo este atenda a chamada. Esta solução permite a saída de qualquer participante a qualquer altura da conferência. A sala só é terminada quando todos os participantes abandonam a conferência.

O segundo cenário suportado pelas *Instant Meeting Rooms* acontece quando os participantes que pretendem telefonar para um DDI, onde são atendidos por um IVR(Interactive Voice Response) do OneContactPBX. Neste cenário, o IVR pede ao participante para marcar o indicador da sala e em seguida faz um pedido, a uma API disponibilizada por o OneMeetingServerManager, a fim de saber para onde deve transferir a chamada. O OneMeetingServerManager, ao receber o pedido, irá devolver a localização da sala com esse indicador ou, se não houver nenhuma em execução, inicia uma sala nesse momento. Neste caso os utilizadores necessitam de combinar previamente o identificador que vão

utilizar para se juntar à sala e a chamada pode ser efetuada por uma extensão interna do OnePBX ou por um telefone pessoal.

A terceira necessidade endereçada pelas *Instant Meeting Rooms* pode ser descrita por os utilizadores poderem iniciar uma videoconferência num dado momento e partilhar um URL, para os participantes WebRTC, ou um identificador, para os participantes SIP. Aqui é utilizado uma aplicação web que faz um pedido a API do OneContactMeetingServer para iniciar uma sala conferência. A API devolve um *token* para ser enviado pela aplicação web e o identificador SIP. Os participantes SIP que pretendem entrar na conferência, procedem aos mesmos passos que os apresentados no segundo cenário, ou seja, digitam o identificador quando telefonam para o IVR do OneContactPBX através de um DDI destinado a esse efeito. Já os utilizadores WebRTC acedem a um link com o *token* da sala e a aplicação inicia a conferência utilizando esse *token* como identificador. Este é um dos cenários que possibilita a coexistência de participantes SIP e WebRTC nas conferências.

Owned Meeting Rooms

As *Owned Meeting Rooms* podem ser temporárias ou permanentes. Estes dois tipos de sala têm que ser previamente provisionadas. As salas permanentes ficam provisionadas enquanto não houver um pedido explícito de eliminação e salas temporárias caducam após serem iniciadas pela primeira vez ou até a uma data de expiração, caso não sejam iniciadas, satisfazendo duas necessidades distintas.

As *Owned Meeting Rooms* permanentes dão ao utilizador uma sala sempre pronta a iniciar e onde o acesso é sempre igual. Assim, o utilizador só necessita de consultar o seu identificador SIP e guardar o link da sua sala apenas uma vez. A sala só pode ser iniciada pelo seu dono. Quando este abandona a conferência pode escolher se pretende desligar todos os outros participantes. Esta é uma funcionalidade importante porque se o utilizador pretender iniciar uma nova conferência na sua sala, esta necessita de estar desimpedida.

Outro cenário suportado pelas *Owned Meeting Rooms* permanentes acontece quando um utilizador combina reuniões periódicas e não pretende partilhar permanentemente o acesso à sala a todos os participantes. Os restantes participantes são informados somente uma vez e ficam com o conhecimento de que o acesso é feito sempre da mesma forma.

As *Owned Meeting Rooms* temporárias suportam um cenário idêntico às *Instant Meeting Rooms*. Os seja, a sala de conferência é criada a pedido, gerando nesse momento um identificador SIP e um link para os participantes WebRTC. A diferença neste caso está no facto de que a conferência não é iniciada no momento do pedido, o que permite aos utilizadores agendar salas de reunião para uma determinada data. A sala expirará um dia depois da data escolhida ou depois de ter sido iniciada. Estas salas também requerem que o seu início seja feito pelo o organizador, o seu dono. Ao contrário das *Owned Meeting Rooms* permanentes, os restantes participantes podem continuar em conferência após o abandono do organizador, sendo esta terminada quando todos os participantes abandonarem a conferência.

3.5 Autenticação

Uma das características das salas é serem públicas. Isto significa que qualquer utilizador que tenha conhecimento do link para aceder a uma determinada sala, pode faze-lo simplesmente carregando esse link no seu *browser*. As salas também possuem um identificador numérico que permite que os utilizadores SIP se possam juntar às salas, fazendo uma chamada para um DDI, atendida por um IVR e procedendo à marcação do identificador quando o IVR o pedir. Esta característica só é válida quando a sala já está em execução, não podendo ser executado o processo de inicialização de uma sala por um utilizador alheio ao sistema.

Para iniciar uma sala de conferências é necessário que um utilizador, dono de uma determinada sala se identifique perante o sistema para que este não possa ser utilizado por utilizadores não autorizados a criar salas de conferência. A solução do OneContact-MeetingServer não fornece qualquer função de gestão de utilizadores nem métodos de autenticação próprios, limitando-se a validar a identificação resultante da autenticação de um utilizador no sistema, a validar *passwords* das salas ou até confiando em pedidos que chegam a determinados *endpoints* privados, dependendo do tipo de utilizadores e cenários em concreto.

Os utilizadores que pretendam iniciar uma conferência através de um *endpoint* WebRTC devem usar as aplicações *web* desenvolvidas para esse efeito. Estas aplicações pedem ao utilizador que proceda ao seu login, caso este seja o dono da sala. O resultado do processo de login, dos utilizadores do OneContactPBX, é um *token* de autenticação através dos protocolos OpenID Connect e Auth 2.0. O OneContactMeetingServer limita-se a validar esse *token*, o qual é enviado na sinalização WebRTC.

Os utilizadores SIP, ao contrário dos WebRTC, não são identificados através de um *token*. Como estes podem estar ligados através de várias interfaces (Hardphones, Softphones, aplicações do OneContactPBX, entre outras), a autenticação é efetuada através de uma *password* somente utilizada para as negociações SIP. Como o OneContactMeetingServer não conhece os utilizadores do OneContactPBX, não é capaz de verificar os utilizadores. Outro problema na identificação dos donos da salas está no facto de que estes podem querer iniciar a sua sala através do seu telefone pessoal, ligando para um DDI público, não passando por qualquer processo de autenticação. Para solucionar este problema, optou-se por todas as salas provisionadas terem uma *password*, somente utilizada para a sua inicialização através de *endpoints* SIP. Esta *password* não está relacionada com nenhum utilizador, podendo qualquer um iniciar uma sala desde que tenha o conhecimento da sua *password*.

As *Instant Meeting Rooms* apresentam um outro cenário em que é complicado garantir que os pedidos de inicialização de salas são feitos por utilizadores autorizados. Neste caso, o OneContactMeetingServer, disponibiliza uma API para os IVRs pedirem o início de uma sala. Esta API deve estar alojada num servidor privado em que somente os componentes dentro da rede tenham o seu acesso. Qualquer pedido, recebido por esta via, é autorizado, tendo como pressuposto que a autorização do utilizador foi verificada anteriormente.

3.6 Persistências dos Dados

A definição da solução de persistências de dados recorreu à análise das soluções disponíveis que melhor satisfizessem os seguintes requisitos (ordenados do mais prioritário ao menos prioritário):

- Gratuito ou tecnologia Microsoft. Por a empresa ser parceira Microsoft e utilizar amplamente as suas soluções, somente tecnologia Microsoft é elegível se for necessário o pagamento de uma licença comercial;
- Mecanismos de redundância;
- Estrutura flexível capaz de comportar facilmente alterações futuras;
- Capaz de comportar as atualizações de produto sem tempos de indisponibilidade;
- Escalável e com boa performance;

- Fácil instalação e manutenção;
- Disponível no Azure no modelo SaaS e ao mesmo tempo ser possível instalar *on-premises* nos clientes.

Baseado nos requisitos funcionais, decidiu-se pela utilização de um SGBD (Sistema de Gestão de Base da Dados) dados NoSQL e orientada a documentos. As base de dados NoSQL têm características mais interessantes, comparativamente com as base de dados relacionais convencionais, que ajudam na resolução de alguns problemas que são apresentados por soluções com a arquitetura e pressupostos do OneContactMeetingServer.

Dos quatro tipos principais de base de dados NoSQL, *document stores*, *graph stores*, *key-value stores* e *wide-column stores*, as orientadas a documentos (*document stores*) são as que melhor respondem às necessidades do OneContactMeetingServer. Estas guardam a informação em forma de documentos facilmente convertidos para o formato JSON e tem uma estrutura dinâmica (*Dynamic Schema*) que facilita a instalação, manutenção e atualização do sistema.

O requisito de *Zero-downtime updates* também influenciou a decisão deste tipo de base de dado, pois este requisito obriga o sistema a funcionar, temporariamente, com componentes em diferentes versões, sendo necessário que o SGBD seja bastante flexível na forma como armazena os dados, deixando para as aplicações a tarefa de suportar a leitura de documentos em diferentes versões.

O SGBD escolhido que melhor responde às necessidades é o MongoDB. Este é orientado a documentos, gratuito, fácil de instalar e manter e com fácil configuração de um *cluster* para redundância dos dados. Também o Azure disponibiliza o serviço Azure Cosmos DB, comercializado no modelo SaaS, e que permite escolher a API em MongoDB, sendo transparente à aplicação estar conectado a uma base de dados MongoDB ou Azure Cosmos DB.

3.7 SIP *Stack*

SIP é um protocolo muito extenso, complexo e muito utilizado para comunicação. Existem inúmeras soluções e *stacks* SIP que ajudam ao desenvolvimento de aplicações que pretendam utilizar o protocolo. Assim este trabalho utiliza uma *stack* externa, pois

seria impossível no tempo disponível desenvolver uma *stack* que implementasse todo o protocolo e tendo em conta toda a oferta disponível não faria sentido esse trabalho.

A Collab já utiliza duas *stacks* SIP noutras soluções desenvolvidas, sendo elas a Radvision e Pjsip. Como a Radvison não apresenta vantagens no desenvolvimento das funcionalidades SIP pretendidas, optou-se pelo Pjsip por ser mais fácil de utilizar e por poder reduzir o tempo de desenvolvimento.

3.8 Síntese

Este capítulo pretendeu introduzir a solução desenvolvida com a descrição dos vários tipos de salas e casos de uso, arquitetura desenvolvida, principais desafios encontrados e métodos de autenticação. Também foram apresentados alguns protocolos de comunicação desenvolvidos, bem como os contextos em que os vários processos comunicam entre si. Por último, tenta mostrar a razão pela escolha de determinadas tecnologias e bibliotecas no desenvolvimento da solução.

No próximo capítulo é dado a conhecer em pormenor cada um dos componentes que constituem o OneContactMeetingServer, em que tecnologias foram desenvolvidos e como eles executam essas tarefas, como por exemplo, a mistura do áudio e gestão das *streams* de vídeo.

4. Descrição e Comunicação entre Módulos

O OneContactMeetingServer, como referido anteriormente, divide-se em três módulos principais, sendo eles a OneMeetingRoom, a OneMeetingPool e o OneMeetingServerManager.

Todos os módulos desempenham tarefas especificas no sistema e todos eles necessitam de comunicar entre si. Neste capítulo irão ser apresentadas as tarefas principais de cada um dos módulos, bem como os diagramas de comunicação que demonstram as interações entre os módulos quando participantes entram num conferência.

4.1 OneMeetingRoom

O OneMeetingRoom é o módulo responsável por suportar uma sala de conferência. Este foi desenvolvido com recurso à linguagem de programação C++ e depende de duas bibliotecas principais, sendo elas a PjSIP e WebRTC. Estas são biblioteca *third party*, também desenvolvidas em C++ e compiladas em conjunto com o restante projeto.

Cada sala de conferência que decorre no sistema é suportada por um processo exclusivo que executa em uma das máquinas disponíveis no sistema e a sua gestão é feita através de um outro processo que suporta a OneMeetingPoool. No final da execução de uma sala, o processo é terminado, não sendo possível o aproveitamento do mesmo processo para suportar novas salas de conferência. O *lifecycle* definido para este processo visa a segurança e proteção contra falhas que são passíveis de acontecer num *software* tão complexo como este.

O OneMeetingRoom é composto por várias camadas lógicas responsáveis por executar tarefas como sinalização, negociação dos canais de media, mistura das *frames* de áudio, distribuição dos vários *streams* de vídeo, entre outras coisas. Estas camadas são

detalhadas de seguida.

4.1.1 Signaling Layer

Quando um participante pretende entrar em uma sala de conferências é necessário que este sinalize a sua intenção à OneMeetingRoom. É também neste processo que é trocada a informação dos medias (áudio/vídeo) utilizando o protocolo SDP. Na OneMeetingRoom, este processo é executado na *Signling Layer*, uma camada lógica que se dedica à troca de mensagem durante um sessão.

Como os participantes que iniciam uma sessão na OneMeetingRoom, podem ser de dois tipos diferentes, SIP e WebRTC, esta camada é dividida em duas partes que comunicam em diferentes protocolos.

Todos os processos de sinalização apresentados nesta secção pretendem demonstrar algumas fases da sinalização na perspetiva da OneMeetingRoom. O processo de sinalização no sistema completo é mais complexo e envolve mais componentes.

SIP Signaling

Como já foi referido anteriormente, o protocolo SIP é um protocolo muito abrangente que, entre outras coisas, define as mensagens trocadas entre duas partes que pretendem iniciar uma sessão. Todo o processo de construção, leitura, manipulação das mensagens é feito pela biblioteca PjSIP.

Na Figura 10 está apresentado a sequência de mensagens trocadas entre a OneMeetingRoom e um participante quando este inicia uma sessão para a sala de conferências.

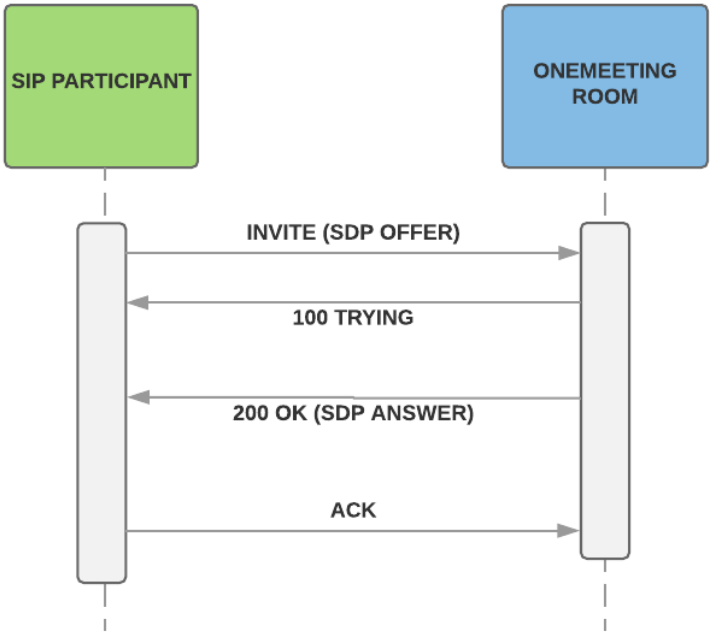

Figura 10: SIP Invite

A mensagem "INVITE"é a mensagem enviada para a sala e que sinaliza a intenção de um participante se juntar a uma conferência. Essa mensagem contém o SDP *offer* com a descrição necessária para a abertura dos canais de media com o participante e que será detalhado posteriormente.

Em SIP existe um mecanismo de retransmissão de mensagens quando num pedido é enviado e não é obtida resposta em um determinado período de tempo. Após algumas retransmissões sem resposta, o pedido é cancelado. Como, por vezes, a resposta ao "INVITE"pode demorar algum tempo, é enviada uma mensagem provisional, opcional, que se destina a informar o requerente que a mensagem foi recebida e está a ser processada (Rosenberg et al., 2002). A OneMeetingRoom, após receber o "INVITE", envia a resposta de código 100 (TRYING) com o objetivo de informar o participante que recebeu o "INVITE"e que está a preparar a sua resposta.

Quando a OneMeetingRoom completa todo o processamento ao pedido, como a preparação dos canais de media, envia a resposta com o código 200 (OK). Esta, para além da confirmação de que o participante foi aceite na sala, leva consigo o SDP *answer* com a informação dos medias necessária para completar a negociação.

Por último, o participante envia a mensagem "ACK". Esta mensagem fecha a transação confirmando que o participante recebeu e processou a informação enviada pela

OneMeetingRoom.

Existe outro cenário possível em SIP, no entanto menos frequente, em que o participante não envia o seu SDP no "INVITE". Quando este cenário ocorre, a sala irá enviar o SDP de *offer* na resposta 200 (OK) e o participante enviará o SDP de *answer* no "ACK". Este cenário é útil em algumas operações originadas pelo *exchange*, como o OnePBX, mas desaconselhado pois a transação é terminada sem confirmação do processamento do SDP de *answer*(Rosenberg et al., 2002).

O processo apresentado pode ser repetido múltiplas vezes durante uma sessão, seguindo exatamente a mesma sequência apresentada, num procedimento designado por "re-INVITE"(Rosenberg et al., 2002). Este procedimento pode ser despoletado pela OneMeetingRoom, pelo participante ou pelo OnePBX, sendo que se a iniciativa for da OneMeetingRoom é invertida a direção das mensagens apresentadas na Figura 10.

4.1.2 WebRTC Signaling

O protocolo escolhido para sinalização WebRTC, embora seja mais simples que o protocolo SIP, a a lógica inerente ao processo de início de sessão semelhante. A grande diferença que se pode encontrar entre estes dois protocolos é o formato das mensagens. Como a tecnologia WebRTC se destina, principalmente, à comunicação entre *Web browsers*, o protocolo é escrito no formato JSON para melhor integração com a linguagem JavaScript.

Este protocolo também foi alterado durante o desenvolvimento do OneContactMeetingServer para endereçar necessidades especificas da OneMeetingRoom, como por exemplo, a lista e informação de todos os participantes presentes na conferência. Na Figura 11 está representado o processo de sinalização utilizado para um participante WebRTC entrar numa sala de conferências.

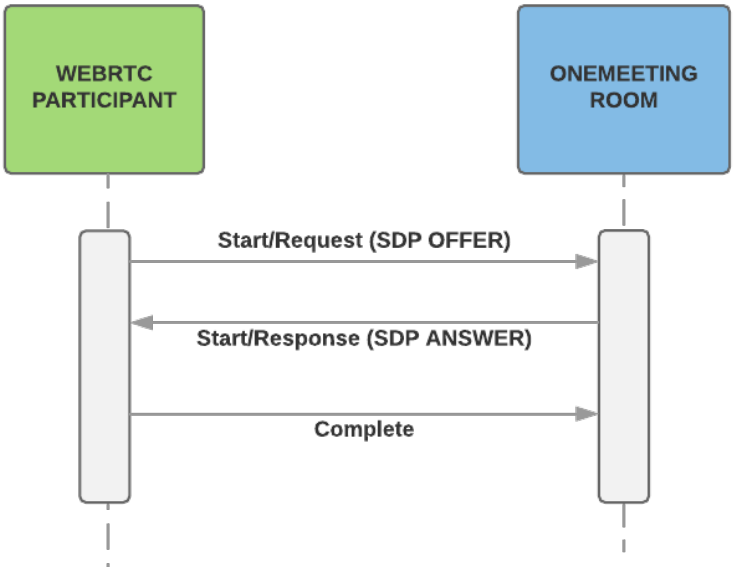

Figura 11: WebRTC Start Transaction

O participante inicia o processo enviando um "Start/Request". Este pedido contém, entre outras coisas, o SDP *offer* gerado pelo *browser* e o nome que o participante pretende mostrar. Ao contrário do protocolo SIP, que embora recomendado não há obrigatoriedade no envio do SDP no primeiro pedido "INVITE", nos participantes WebRTC o envio SDP de *offer* é obrigatório nesta mensagem.

Após a sala processar o pedido com a descrição dos medias do participante, esta envia a resposta de confirmação ao participante, o "Start/Response". Esta mensagem, para além do SDP de *answer*, leva consigo a lista com os identificadores de todos os participantes. Esta lista pretende fornecer informação, a cada participante, de todos os outros que nela participam. Os SDP's utilizados para a comunicação WebRTC geralmente ocupam muito espaço. Por isso, para reduzir o tamanho das mensagens, a lista apenas contém os identificadores sendo necessário o envio de outro pedido para obtenção da informação detalhada de cada participante.

A mensagem "Complete"destina-se somente a informar a sala de que o participante já processou a resposta, finalizando a transação.

Quando um participante pretende conhecer a informação detalhada de todos os participantes, este pode iniciar um nova transação que consiste somente em um pedido e uma resposta. Este pedido, o "Info/Request", pode conter a lista de participantes sobre a qual se pretende ter informação ou ir vazio caso se pretenda a informação de todos os participantes. Em resposta, a OneMeetingRoom envia a mensagem "Info/-

Response"contendo o identificador do participante, o nome do participante ou número de telefone, caso seja um participante SIP, e se está a participar com vídeo ou apenas áudio.

Do mesmo modo que o processo de "re-INVITE"acontece em SIP, em WebRTC estas mensagens podem também ser trocadas múltiplas vezes durante uma sessão. Neste tipo de participantes, a sala inicia essas mensagens sempre que um novo participante se junta a conferência com vídeo, pois todos os participantes WebRTC recebem uma *stream* de vídeo por cada participante e é necessária a troca de um novo SDP onde conste a informação do vídeo do novo participante.

4.1.3 Canais de Media

Para que os participantes possam receber e enviar áudio e vídeo é necessário a troca de informação sobre a forma como esses *medias* vão ser trocados. A negociação entre os participantes da OneMeetingRoom apresenta algumas diferenças entre os participantes SIP e os participantes WebRTC. Embora ambos façam uso do protocolo SDP para proceder à negociação, o resultado final é diferente e os SDP gerados por os participantes WebRTC não são compatíveis com os participantes SIP.

Como o protocolo SIP é um protocolo com um nível de maturidade muito elevado e amplamente utilizado pela indústria de telecomunicações por vários fabricantes, os SDP utilizados não podem divergir dos *standards* e seguem o RFC-4566 para garantir a interoperabilidade dos seus produtos. O WebRTC, ao contrário do SIP, é um protocolo muito recente, que está em constante atualização e tem uma utilização menor que o protocolo SIP. O WebRTC implementa funcionalidades de segurança e de eficiência, que não estão contemplados no RFC-4566, usando alguns campos que são encontrados em drafts e RFCs de extensão ao RFC-4566. Também diferentes *browsers* usam diferentes versões de SDP, por exemplo, atualmente um SDP com múltiplas *streams* de vídeo gerado pelo Google Chrome não é compatível com um SDP gerado pelo Mozilla Firefox.

Para o desenvolvimento deste projeto, a Collab decidiu que a prioridade é a interoperabilidade com o *browser* Google Chrome e que o Mozilla Firefox iria receber suporte mais tarde. Com base neste pressuposto, todo o desenvolvimento feito só teve em consideração o Google Chrome.

Canais Media SIP

Como já foi demonstrado na secção anterior, para iniciar uma chamada para uma OneMeetingRoom é necessário a troca de mensagens SIP especificas para o efeito. Essas mensagens além de informar a OneMeetingRoom que um participante pretende juntar-se à conferência, transportam também o SDP com a descrição necessária para abertura dos canais. Na Figura 12 está um exemplo de um SDP recebido pela OneMeetingRoom e a respetiva explicação quando um participante entra na conferência.

```
1  v=0
2  o=- 3 2 IN IP4 127.0.0.1
3  s=CounterPath eyeBeam 1.5
4  c=IN IP4 127.0.0.1 1
5  t=0 0
6  m=audio 4696 RTP/AVP 100 106 6 0 97 105 98 8 18 3 5 101
7  a=fmtp:18 annexb=yes
8  a=fmtp:101 0-15
9  a=rtpmap:100 SPEEX/16000
10 a=rtpmap:106 SPEEX-FEC/16000
11 a=rtpmap:97 SPEEX/8000
12 a=rtpmap:105 SPEEX-FEC/8000
13 a=rtpmap:98 iLBC/8000
14 a=rtpmap:18 G729/8000
15 a=rtpmap:101 telephone-event/8000 6
16 a=sendrecv 7
17 a=x-rtp-session-id:C7749C3040854C7A97DE267EB0DB5BF7
```

Figura 12: SIP SDP Offer

- **1:** IP do participante.
- **2:**. Tipo de media que é descrito na secção (pode conter várias secções com diferentes tipos de media).
- **3:** Porto que o participante está à escuta do media.
- **4:** Protocolo de transporte.
- **5:** Lista de *codecs* suportados.
- **6:** Descrição dos *codecs* suportados. Cada linha tem o código de um *codec* da lista e a sua descrição. Nem todos os *codecs* estão descritos porque alguns tem um código fixo, dispensado a sua descrição (por exemplo, o código 0 é sempre respetivo ao PCMU/8000 e o 8 ao PCMA/8000).

- **7:** Sentido da *stream*. Neste caso a *stream* é bidirecional (envia e recebe áudio).

Quando o SDP apresentado é processado, a OneMeetingRoom escreve um SDP de *answer* que é enviado na resposta SIP 200 (OK). Esse SDP é semelhante ao enviado pelo participante, apresentado na Figura 12, com a diferença de que contém apenas o *codec* escolhido da lista de *codecs* enviada pelo participante. Como a sala é a última a responder, fica a conhecer os *codecs* suportados pelo participante e é ela a responsável pela escolha do *codec* que irá ser utilizado na comunicação.

Nos clientes SI, todos os canais de media são UDP. O endereço IP e o porto escolhidos por cada uma das partes (participante e OneMeetingRoom) são utilizados para envio e receção dos pacotes de media. No exemplo apresentado anteriormente, o participante e a sala trocam pacotes RTP em UDP com a informação do áudio no formato do *codec* escolhido e enviados para o endereço IP e porto lidos do SDP recebido.

Os participantes SIP podem também utilizar o protocolo SRTP (Secure Real-time Transport Protocol) em substituição ao RTP. O pacote SRTP é somente um pacote RTP encriptado, conferindo segurança como a confidencialidade e integridade dos pacotes RTP(Baugher et al., 2004).

A OneMeetingRoom somente suporta canais de media em UDP para os participantes SIP.

Dificuldades de NAT

O método de NAT permite o uso do mesmo endereço IP por várias máquinas inseridas numa rede privada. Esta técnica resolve o problema da exaustão de endereços IPv4 e, por isso, é globalmente utilizada. Todas as redes privadas, como as redes domésticas ou profissionais, que se pretenda conectar à Internet, usam este protocolo de mapeamento para traduzir o seu IP e porto privado pelo IP e porto público. Existem ainda vários tipo de NAT, diferenciando a forma como o mapeamento é feito, que em conjunto com *Firewalls* causam algumas dificuldades às ligações *peer-to-peer*.

Quando um participante está numa rede privada e pretende entrar numa sala de conferência, a máquina que faz o pedido não conhece para quais IP e porto o seu IP e porto privado serão mapeados. O participante ao preencher o seu SDP sobre os canais de *media* que preparou está a fornecer um IP e porto inacessível para quem está fora da

rede, ficando indisponível para receber os pacotes de *media* das OneMeetingRooms.

Existem alguns protocolos que visam a resolução deste problema, como o protocolo ICE. Embora o protocolo SDP esteja preparado para integrar com o protocolo ICE, este necessita de algumas configurações por parte dos clientes, como configuração de servidores STUN (Session Traversal Utilities for NAT)(Rosenberg, 2010a).

STUN é um protocolo utilizado como ferramenta para outros protocolos, como o SIP, conseguirem determinar o endereço IP e o porto alocado pelo NAT. Para alguns tipos de NAT, como o NAT assimétrico, as configuração de servidores STUN são insuficientes, necessitando também de configurar um servidor TURN (Traversal Using Relays around NAT)(J. Rosenberg and Wing, 2008)(R. Mahy and Rosenberg, 2010).

TURN é um protocolo que faz uso de servidores como *relayers* para resolver o problema da comunicação. Assim, a OneMeetingRoom transmite todos os pacotes de *media* para o servidor de *relay* que retransmite os pacotes para o cliente(R. Mahy and Rosenberg, 2010).

Embora o protocolo ICE resolva o problema de NAT, e esteja integrada na tecnologia WebRTC, o uso deste protocolo pela OneMeetingRoom para os participantes SIP pressupunha o suporte do protocolo por todos os clientes SIP, a sua configuração em todos os clientes e a existência dos servidores TURN e STUN. Todos estes pressupostos dificultam a utilização das salas de conferência o que afastou a sua utilização.

A OneMeetingRoom para resolver o problema apresentado, implementa uma técnica de descoberta do endereço IP e do porto durante toda a troca de pacotes media. Como a OneMeetingRoom não está atrás do mecanismo de NAT, esta consegue sempre conhecer o seu IP e porto público, conseguindo garantir a receção dos pacotes dos clientes. Para garantir que os clientes consigam receber os seu pacotes, o porto é exclusivamente alocado a um participante, conseguindo desta forma mapear os pacotes recebidos ao respetivo participante. Para cada pacote de *media* recebido pela OneMeetingRoom é lida a sua a origem. Quando são recebidos mais de dez pacotes com origem diferente ao endereço IP e porto para onde os pacotes estão a ser enviados, a OneMeetingRoom altera automaticamente o endereço e porto do destino.

A técnica utilizada, para além de garantir a correta passagem dos pacotes pelo mecanismo de NAT, possibilita que um participante continue a receber os medias caso haja alterações de rede, como mudanças de rede ou mudança de IP público. Como a OneMeetingRoom está continuamente a verificar o IP e porto de origem, esta detetará

sempre as alterações e consequentemente alterará o IP e porto de destino.

Canais Media WebRTC

Os canais de *media* dos participantes WebRTC são substancialmente diferentes dos canais dos participantes SIP. As maiores diferenças são a encriptação dos canais, uso da técnica de *multiplexing* e a diferente forma de como o endereço IP e porto são trocados no SDP.

O *browser* Google Chrome, escolhido para ser o primeiro suportado pelo OneContact-MeetingServer, faz uso de muitas extensões ao SDP, definido no RFC-4566. Algumas extensões estão também definidas em outros RFCs e outras ainda são só propostas e constam em documentos *draft*.

Um dos desafios apresentados no capítulo anterior é o problema do número limitado de portos disponíveis numa máquina. Se cada *stream*, de áudio ou vídeo, requeresse a alocação de dois portos exclusivos na OneMeetingRoom, uma para o RTP e outro para o RTCP, como acontece nos participantes SIP, rapidamente esgotaríamos os portos disponíveis. Cada participante, alocaria dois portos para áudio, dois para a *stream* de vídeo principal, mais dois por cada *stream* dos outros participantes, tendo uma taxa de crescimento de $2 * (N + 1) * N$. O WebRTC já resolve o problema com a técnica de *multiplexing*, necessitando apenas um porto por participante. Esta técnica consiste na transformação de múltiplos sinais em um só sinal e está apresentada na Figura 13.

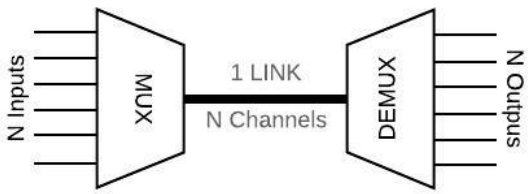

Figura 13: Multiplexing

Outra diferença que os canais WebRTC apresentam em relação aos canais SIP é a encriptação obrigatória. Enquanto os canais SIP, suportados pelo OneMeetingServer, são UDP e trocam pacotes RTP ou SRTP, sendo que o uso de SRTP é pouco frequente, o WebRTC troca pacotes RTP sobre DTLS(Datagram Transport Layer Security) em canais TCP ou UDP, embora o uso de UDP seja preferido para comunicações em *real-time*.

O WebRTC também usa o protocolo ICE, referido anteriormente. Como um dos objetivos do protocolo é a obtenção de possíveis endereços IP e porto públicos, este, através do protocolo STUN, forma uma lista de endereços e portos candidatos que é enviada no SDP.

Para melhor compreender tudo o que é necessário na negociação dos canais de media WebRTC apresenta-se na Figura 14 um exemplo de um SDP e em baixo a respetiva explicação.

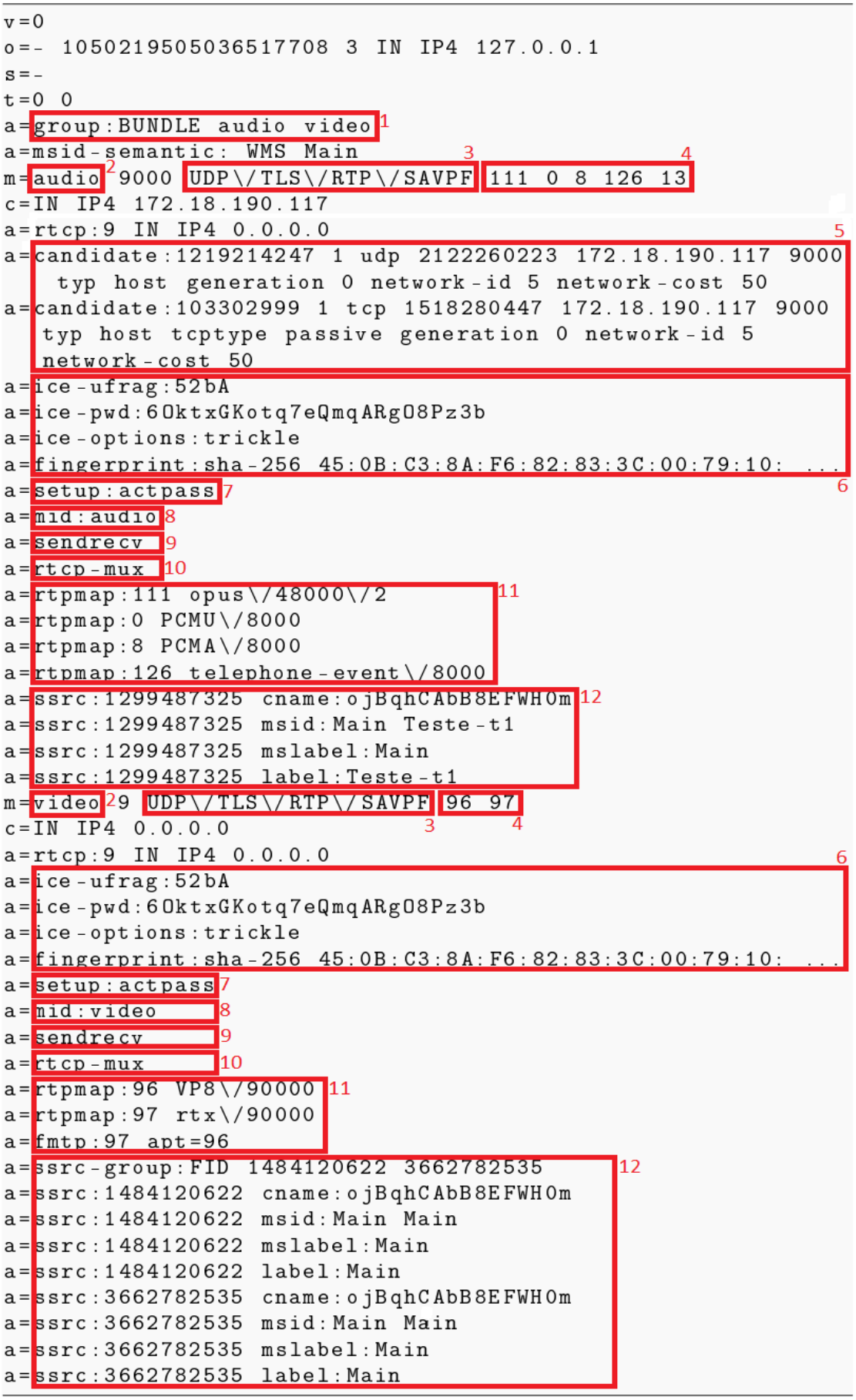

Figura 14: WebRTC SDP Offer

- **1:** Criação de um grupo de dois *medias* com o id "Audio"e outro "Video"que indicam o uso da técnica de multiplexing(Holmberg et al., 2017).
- **2:** Tipo de *media*.
- **3:** Protocolo de transmissão utilizado, neste caso RTP sobre DTLS(McGrew and Rescorla, 2010).
- **4:** Lista de *codecs* suportados.
- **5:** Lista de candidatos(Rosenberg, 2010b).
- **6:** Atributos de encriptação dos candidatos(Rosenberg, 2010b).
- **7:** Sentido da negociação DTLS(Yon and Camarillo, 2005)(Fischl et al., 2010)(Holmberg and Shpount, 2017).
- **8:** Id do media no Bundle, escrito no ponto 1(Camarillo and Schulzrinne, 2010)(Holmberg et al., 2017).
- **9:** Sentido do *stream*, no caso destes dois medias, o *stream* é bidirecional.
- **10:** Usar o mesmo canal de *multiplexing* para envio do RTCP(Perkins and Westerlund, 2010)(Holmberg et al., 2017).
- **11:** Descrição dos codecs que foram apresentados na lista.
- **12:** Este é um atributo utilizado especificamente pelo Google Chrome e destina-se à identificação de vários *streams* de vídeo. Independentemente de o número de *streams* que é negociado para um determinado tipo de *media*, somente é escrita uma linha sobre esse media e aqui é adicionado mais uma *source*. Esta técnica é conhecida com *Plan B* e somente existe um documento *draft* expirado sobre o assunto(Uberti, 2013).

4.1.4 Audio Engine

Para ser possível a conferência de áudio é necessário um módulo que faça a gestão de todas as *streams* de áudios dos participantes e que as misture. O *Audio Engine* é uma camada lógica dentro da OneMeetingRoom encarregue deste processo.

O *Audio Engine* é composto por dois elementos principais, o *Audio Mixer* e o *Audio Stream Manager*. O *Audio Mixer* é responsável pela mistura das *frames* de áudio,

resampling e identificação do participante que está a falar em cada momento. O *Audio Stream Manager* é a camada que faz a gestão das *streams* de áudio dos participantes SIP e WebRTC e faz a abstração das *streams* para o *Audio Mixer*, deixando-o sem a perceção de que tipo de participante é uma determinada *stream*.

Audio Mixer

Para haver áudio no formato digital é necessário um processo de discretização (*sampling*) resultando um determinado número de amostras que varia consoante a frequência escolhida. Para se juntar dois ou mais *streams* é necessário que estas se encontrem na mesma frequência, pois a soma das amostras de tempos diferentes resulta em um som deficiente.

A *stream* adicionada ao *Auido Mixer* é uma *stream* lógica, gerada pelo *Audio Stream Manager* e apresenta as seguintes parâmetros:

- **Frequência**. Frequência em Hz da stream. Número de amostras por segundo.
- **Ptime**. Intervalo de *frames*. Os pacotes de áudio recebidos e enviados são correspondentes a um intervalo de tempo, o Ptime. Em alguns *codecs*, como o ILBC, o Ptime é configurável.
- **Canais**. Número de canais. Podem ser utilizados um (mono) ou dois (stereo) canais de áudio. Numa comunicação de telefone, tipicamente é somente usada um canal.

Para além das configurações, esta *stream* lógica disponibiliza duas ações, uma utilizada para o misturador pedir a próxima *frame* de áudio e a outra para enviar a *frame* produzida para o participante.

Quando as *streams* de áudio chegam ao servidor, chegam em um determinado *encoding* definido pelo *codec* negociado. Embora as *streams* processadas pelo *Audio Mixer* já tenham sofrido o processo de *decoding*, encontrando-se no formato PCM (Pulse-code modulation), a frequência é definida pelo o *codec* negociado entre o participante e a OneMeetingRoom.

Para que o *Audio Mixer* misture corretamente *streams* que podem apresentar diferentes parâmetros, também ele é configurado com uma frequência, Ptime e número de canais,

que servem como base para a mistura das *frames*. Todas as *streams* com frequências diferentes à do *Audio Mixer* sofrerão o processo de *resampling*, transformando-as para a mesma frequência que o misturador. Também as *streams* com diferente número de canais, tem que ser convertidas para o número de canais do misturador.

O Ptime define o *clock* da mistura, por exemplo, se o Ptime for 20 milissegundos, o *Audio Mixer* executa a mistura de *frames* de 20 em 20 milissegundos.

Embora possam ser alterados, os valores por defeito do *Audio Mixer* são 8kHz mono com 20 milissegundos de Ptime. Estas configurações visam a poupança de recursos, principalmente CPU, garantido boa qualidade dentro dos padrões esperados numa comunicação em *real-time*. A frequência de 8kHz é a frequência mínima utilizada pelos *codecs* suportados pela OneMeetingRoom. Embora quanto menor é a frequência, menor é a qualidade do som, o valor de 8kHz mostrou ser suficiente para assegurar a qualidade pretendida e diminui o número de amostras a ser processado a cada *frame*, comparativamente com outras frequências. Em relação ao Ptime, quanto maior for o seu valor, maior é o atraso do som, sendo necessário arranjar uma boa relação entre o atraso e o consumo de processador. O valor de 20 milissegundos é o valor mais utilizado neste tipo de comunicações e é o valor utilizado pela maioria dos *codecs*. Por último, a utilização de áudio em mono é o mais indicado para estes casos. A grande maioria dos *codecs*, suportado pelo OneMeetingRoom só lidam com áudio em mono, não existindo justificação para configurar o *Audio Mixer* com mais do que um canal.

O *Áuido Mixer* necessita de executar alguns cálculos para perceber quantas amostras necessita de processar para cada *stream* e quantas vai processar a cada ciclo. Em seguida são apresentadas as fórmulas utilizadas pelo *Audio Mixer* e as variáveis utilizadas são:

- **Af**. Frequência em Hz utilizada pelo *Audio Mixer*.
- **Apt**. Ptime utilizado pelo *Audio Mixer*.
- **Ac**. Número de canais utilizado pelo *Audio Mixer*.
- **As**. Número de amostras por *frame* do *Audio Mixer*.
- **Sf**. Frequência em Hz de uma *stream*.
- **Spt**. Ptime de uma *stream*.
- **Sc**. Número de canais de uma *stream*.

- **Ss**. Número de amostras por *frame* de uma *stream*.
- **SAs**. Número de amostras de uma *stream* que completam uma *frame* do *Áudio Mixer*.

Para o *Audio Mixer* calcular o número de amostras que vai processar a cada ciclo (As) é utilizada a seguinte fórmula:

$$As = \frac{Af * Apt * Ac}{1000}$$

O cálculo do número de amostras que forma uma *frame* de uma *stream* (Ss) é descrito pela seguinte fórmula:

$$Ss = \frac{Sf * Spt * Sc}{1000}$$

Também é necessário calcular o número de amostras a ser lido em cada *stream* para completar uma frame do *Audio Mixer* (antes do *resampling*). A fórmula utilizada é a seguinte:

$$SAs = As * Sc * \frac{Sf}{Af}$$

É necessário ter em consideração que o Ptime definido para o *Audio Mixer* pode ser diferente do Ptime da *stream*. Assim o *Audio Mixer* pode necessitar pedir de mais que uma *frame* para completar o número de amostras calculado para Ss ou guardar amostras para o próximo ciclo. Para exemplificar, o *Audio Mixer* com os valores por defeito, 8kHz mono com 20 milissegundos de Ptime recebe uma *stream* com 16kHz mono com 10 milissegundos de Ptime. Neste exemplo, o $As = 160$, $Ss = 160$ e $SAs = 320$, sendo necessário que o *Audio Mixer* leia duas *frames* da *stream* para completar uma *frame* do *Audio Mixer*. Os cálculos dos valores apresentados são:

$$As = \frac{8000 * 20 * 1}{1000} = 160$$
$$Ss = \frac{16000 * 10 * 1}{1000} = 160$$

$$SAs = 160 * 1 * \frac{16000}{8000} = 320$$

Continuando o exemplo apresentado concluímos que para 20 milissegundos de áudio a *stream* contém 320 amostras. Como o *Audio Mixer* está configurado para processar 160 amostras por cada 20 milissegundos, é necessário fazer o *resampling* da *stream* para o áudio ser representado com a mesma *sample rate* que a do *Audio Mixer*.

O processo de leitura e preparação das *frames* das *streams* a serem utilizadas para fazer a mistura está representado na Figura 15.

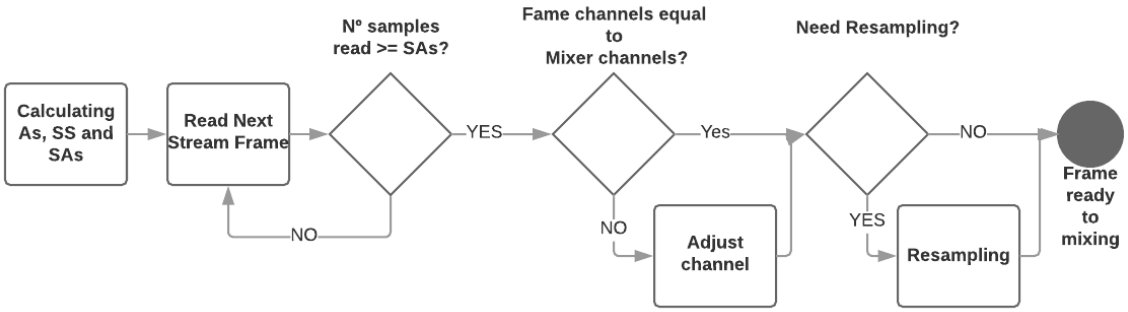

Figura 15: Processo de leitura e preparação das frames de áudio

Redução de ruído

Todas as *streams* de áudio têm ruído, esse ruído pode ser sons de fundo ou introduzidos pelos dispositivos de captura de áudio. Também o processo de *resampling* pode causar a adição de pequenos artefactos que originalmente não existem. Esse ruído não é relevante quando duas pessoas estão numa chamada. Numa conferência de múltiplos participantes, o ruído de todos os participantes é adicionado à *stream* de saída, quanto maior o número de participantes, maior é o ruído.

Para reduzir o ruído da conferência foi implementado um algoritmo de redução de ruído para conferência. Este é um algoritmo muito básico mas que tem bons resultados com pouca afetação no áudio necessário à conferência. O algoritmo consiste em, cada frame do *Audio Mixer*, procurar os três participantes mais ativos e ignorar o áudio dos outros participantes. Como esta escolha é feita a cada frame, por defeito são 20 milissegundos de áudio, esta não afeta a conversa dos participante, pois uma frame com mais de três participantes com um grande nível de atividade também irá resultar em ruído.

Mixing

Como já foi referido anteriormente, o *Audio Mixer* junta *frames* de áudio em formato PCM. Depois das *frames* de cada participante terem sido calculadas e de escolhidos as três *frames* mais ativas, o *Audio Mixer* soma as amostras dessas *frames*. Com as três *frames* com o mesmo *sample rate*, resultantes do processo anteriormente demonstrado, o *Audio Mixer* limita-se a proceder a uma soma de vetores em que cada vetor é uma *frame* e cada posição é uma amostra dessa *frame*. O resultado é uma *frame* com os três áudio misturados. O processo de mistura está representado na Figura 16.

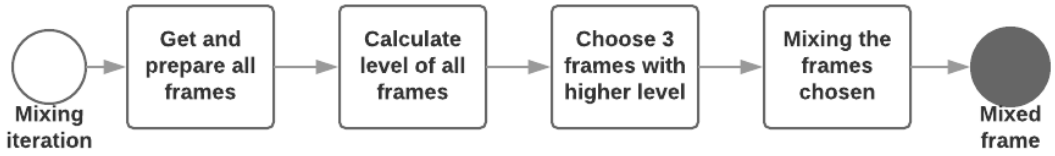

Figura 16: Processo de mistura do áudio

Depois da *frame* misturada calculada, inicia o processo de envio. Este apresenta mais passos que os anteriores e está representado na Figura 17.

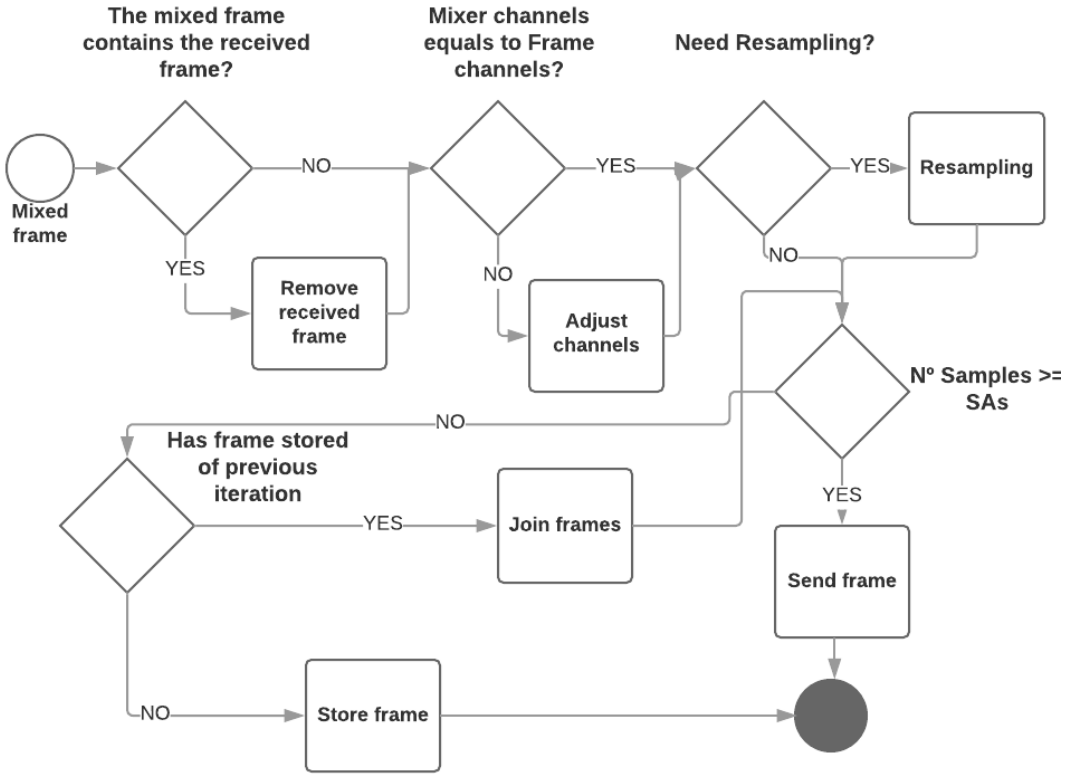

Figura 17: Processo de preparação da frame de áudio misturada a enviar

Se um determinado participante foi escolhido para construir a *frame* misturada é necessário a subtração do seu áudio antes de ser enviado. Se não for retirado o áudio do próprio participante, este ouve o seu próprio áudio dificultando a comunicação. Para proceder a esta tarefa é necessário guardar a *frame* de cada participantes, saber se foi utilizada na produção da *frame* misturada e em seguida fazer uma subtração de vetores. Aqui é subtraído ao vetor, com o áudio misturado, o vetor de entrada desse participante resultando uma *frame* final com o áudio dos outros dois participantes. Os restantes participantes que não contribuíram para a mistura recebem a *frame* misturada sem alterações.

Quando a *frame* final de um participante estiver calculada, é iniciado o processo inverso ao da Figura 15. Este processo pretende transformar a *frame*, caso necessário, para o *sample rate* e número de canais negociado com o participante. Se depois da transformação da *frame* o número de amostras for menor a Ss (o número de amostras de uma *frame* da *stream* do participante), que acontece sempre que o Ptime do *Audio Mixer* é menor que o Ptime da *stream*, é necessário aguardar pela próxima *frame* gerada para completar a *frame* da *stream*.

4.1.5 Vídeo Engine

Os participantes WebRTC são os únicos que conseguem participar numa conferência nesta versão. Embora esteja planeado os participantes SIP poderem enviar e receber vídeo, esta funcionalidade não é prioritária para a Collab, pois não existe atualmente nenhum cliente que tenha este requisito.

Cada participante WebRTC pode receber até 9 *streams* de vídeo, uma *stream* principal que corresponde ao participante que está a falar em cada momento e as secundárias que correspondem a cada um dos participantes presentes com vídeo. Como o processamento de vídeo ainda é exigente para as máquinas dos clientes e por não ser muito útil a visualização das *streams* de todos os participantes quando estes já representam um número considerável, decidiu-se limitar a visualização fixa até 8 participantes.

Video Engine é a camada lógica que trata da gestão das *streams* de vídeo, ao contrário das *streams* de áudio que necessitam de ser misturadas, as *streams* de vídeo são enviadas isoladamente para cada participante. Comparativamente com o *Audio Engine*, o *Video Engine*, desenvolvido nesta fase, é um módulo mais simples por não necessitar de alterar as *frames* de vídeo. Contudo, o processamento de *streams* de vídeo requerem

mais recursos das máquinas sendo necessária a sua otimização no futuro, tornando o *Video Engine* um módulo muito mais complexo.

Cada vez que um novo participante de vídeo se junta à conferência e caso o limite de *streams* recebidas pelos restantes ainda não tenha sido atingido, esta nova *stream* adicionada ao *Video Engine* faz despoletar um evento para a *Signaling Layer*, para esta iniciar uma nova transação na sinalização com o objetivo de informar no SDP o aparecimento desta nova *stream*. Se a OneMeetingRoom não executar o processo de sinalização e iniciar o envio desta nova *stream* pelo o canal já criado no envio das outras *streams*, os pacotes correspondentes a esta nova *source* são ignorados pelo cliente.

Os participantes de vídeo podem falar em diferentes *codecs*, embora atualmente só seja suportado o Google Chrome e seja fácil garantir que todos falam no mesmo *codec*, no futuro com a integração de outros *browsers* e integração de vídeo nos *endpoints* SIP, a condição de que todos participam com o mesmo *codec* pode ser difícil de garantir. Para assegurar a correta transmissão das *streams* de vídeo, o *Video Engine* faz sempre o processo de *transcoding*.

O processo de *transcoding* consiste na receção da *stream* de vídeo, fazer o *decoding* para ficar com as *frames* de vídeos descomprimidas e voltar a fazer o *encoding* com o *codec* do destinatário da *stream*. Este processo está representado na Figura18.

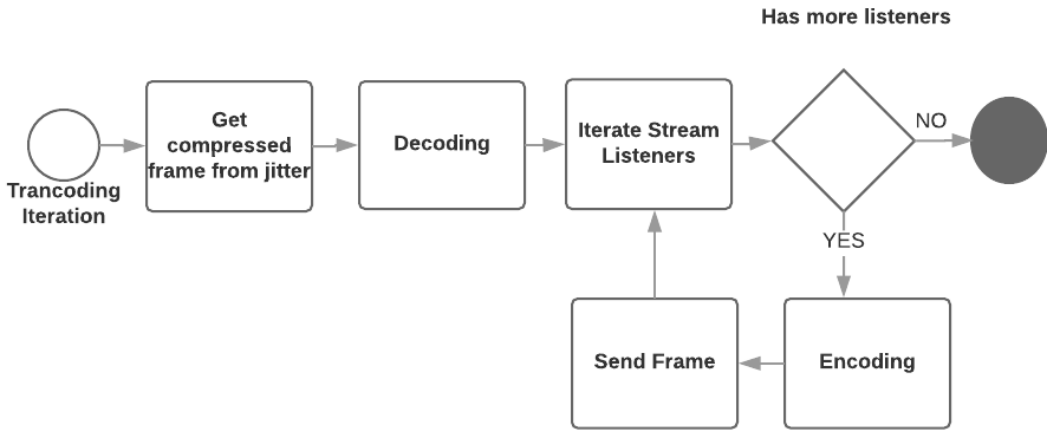

Figura 18: Processo de trancoding da stream de vídeo

4.2 OneMeetingServerManager

O OneMeetingServerManager é o processo responsável por gerir todo o sistema. Este foi desenvolvido em C# com recurso a .NET Standard 2.0, permitindo assim a possibilidade de ser executado no sistema operativo Windows como um serviço Windows em .NET Framework 4.6.1 ou superior e como uma aplicação de consola em .NET Core 2.0 ou superior capaz de executar em sistemas operativos Windows ou Linux.

É o OneMeetingServerManager que faz a distribuição das salas de conferência pelas várias máquinas disponíveis nos sistema e, por isso, é a este que é feito o pedido para iniciar uma nova sala. O OneMeetingServerManager disponibiliza várias interfaces de comunicação para a realização das operações de inicialização e consulta das salas e também está permanentemente à espera de novos registos de OneMeetingPools, não sendo necessário qualquer configuração adicional para receber mais uma máquina no sistema.

Em um sistema completo em que se pretende *Zero down times* do serviço, devem ser instalados dois processos destes em máquinas distintas e recorrer a um mecanismo de *Failouver Clustering*. Este mecanismo, tem por objetivo assegurar a continuidade do serviço quando uma máquina deixa de responder, encaminhando todos os pedidos subsequentes para a outra instância que assume a tarefa de gestão do sistema.

O OneMeetingServerManager pode ser divido em três componentes lógicos, sendo eles o OneMeetingServerInfoPoint, OneMeetingServerPoolManager e o OneMeetingServerOperator.

OneMeetingServerInfoPoint

O OneMeetingServerInfoPoint é uma camada lógica que se destina a receber pedidos e enviar respostas sobre a informação das salas. Ainda, também é o ponto de entrada para os pedidos de iniciação de sala. As informações devolvidas nas respostas correspondem à localização da sala requerida, através da combinação de IP/Porto dos diferentes *endpoints* suportados pela sala.

Esta camada pode expor dois tipos diferentes de comunicação para melhor integração dos componentes externos.

O primeiro tipo de comunicação utilizada é mensagens em Protocol Bufferes sobre *Sockets* TCP e utilizando o OneMeeting Message Protocol. Esta é a comunicação utilizada pelas OneMeetingWebrtcServices e pela Runtime API. Como este tipo de comunicação apresenta algum nível de complexidad, foi desenvolvido também uma biblioteca em .NET Standard 2.0 que serve de cliente do OneMeetingInfoPoint e que encapsula a complexidade da comunicação. Esta biblioteca pode ser facilmente utilizada por componentes externos, como é o caso dos IVRs.

A outra forma de comunicação é através de uma Gateway SIP. Esta é unicamente utilizada pelo OnePBX para os pedidos referentes às *Instant Meeting Rooms* que suportam o cenários da criação de salas de conferência instantâneas, pelas as aplicações cliente do OneContactPBX, quando estas estão em uma chamada de duas pessoas e pretendem adicionar outros participantes à conversa.

Esta camada trata unicamente da comunicação, não tendo qualquer tipo de lógica para tomada de decisões. Deste modo, é iniciada com a referência de um OneMeetingServerOperator, detalhado posteriormente, e todos os pedidos recebidos são encaminhados para esse componente.

Esta camada pode também ser separada do processo do OneMeetingServerManager e instalada separadamente em outra máquina se for necessário implementar o balanceamento de carga.

OneMeetingServerPoolManager

O OneMeetingServerPoolManager é a camada que trata da gestão de todas as OneMeetingPools do sistema. É esta camada que recebe o registo das OneMeetingPools e trata de todas as interações como elas. Também é esta a camada responsável pelo balanceamento da carga no sistema, pois é ela que decide em que OneMeetingPool uma determinada sala irá executar.

OneMeetingServerOperator

O OneMeetingServerOperator é onde reside a lógica de iniciação da sala. Este é responsável por receber os pedidos de início, fazer as validações necessárias e guardar a informação das salas iniciadas. É também responsável pela comunicação com o One-

MeetingServerPoolManager quando é necessário iniciar uma nova sala.

No caso do OneMeetingServerInfoPoint ser instalado separadamente do processo do OneMeetingServerManager, o OneMeetingServerOperator poderá também substituir a comunicação com o OneMeetingServerPoolManager pela comunicação com OneMeetingServerInfoPoint que está copulado ao OneMeetingServerManager. Neste caso, o OneMeetingServerOperator faz o mesmo trabalho quando trata pedidos às salas já iniciadas mas delega a função de início de sala para o OneMeetingServerOperator do OneMeetingServerManager, pois esta é a única camada capaz de comunicar com o OneMeetingServerPoolManager.

4.3 OneMeetingPool

A OneMeetingPool é o processo que faz a gestão das OneMeetingRooms numa determinada máquina. Esta foi desenvolvida usando as mesmas tecnologias que as utilizadas no OneMeetingServerManager, C# .NET Standard 2.0. Este processo pode ser executado em sistemas operativos Windows e Linux da mesma forma que o anterior. Não há uma obrigatoriedade das OneMeetingPools instaladas num sistema estarem a executar no mesmo sistema operativo que o OneMeetingServerManager a que se vai conectar.

As OneMeetingPools registam-se no OneMeetingServerManager e ficam à espera que este as notifique para iniciarem uma OneMeetingRoom e devolvem uma resposta com a informação dos *endpoints* SIP e WebRTC da OneMeetingRoom iniciada.

Também as OneMeetingPools estão encarregues de gerir as configurações das OneMeetingRooms. Como são as OneMeetingPools que iniciam os processos das OneMeetingRooms, que necessitam de inúmeras configurações incluindo portos reservados, é nas OneMeetingPools onde todas as configurações são definidas. Para iniciar um OneMeetingPool com sucesso são necessárias as seguintes configurações:

- **OneMeetingServerManager *Endpoint***. Esta configuração é a única que não está relacionada com as OneMeetingRooms e destina-se apenas à configuração do endereço IP e do porto onde o OneMeetingServerManager está à escuta das conexões das OneMeetingPools;
- **IP interno**. Este é o endereço IP privado da máquina. A utilização deste endereço depende de outras configurações.

- **IP Externo**. Este é o endereço IP público da máquina e é utilizado na negociação dos canais de media, através do SDP, para os participantes SIP e WebRTC. Este pode ainda ser utilizado para outras finalidades dependendo de outras configurações.

- **Portos reservados para a sinalização SIP em TCP**. Cada OneMeetingRoom necessita de um porto para receber a sinalização SIP em TCP, como cada porto só pode ser utilizado exclusivamente por um processo, esta coleção é utilizada para a OneMeetingPool distribuir os portos pelas OneMeetingRooms sem causar colisões.

- **Portos reservados para a sinalização SIP em UDP** Esta configuração é semelhante à anterior com a diferença de ser utilizada para sinalição SIP em UDP. Esta coleção geralmente é igual à anterior, pois os sistemas operativos permitem abrir um porto UDP com o mesmo número de um porto TCP.

- **Usar IP Externo para Sinalização SIP**. Esta é uma configuração de valor booleano e que define se o IP utilizado para sinalização é o IP público ou privado. Esta configuração deve ter sempre o valor "*false*"(valor por defeito), pois a OneMeetingRoom não tem qualquer tipo de validação dos participantes, sendo esta responsabilidade do OneMeetingServerManager e do OnePBX. O IP escolhido deve ser acessível apenas pelo OnePBX e toda a sinalização SIP deverá ser feita exclusivamente através dele.

- **Portos reservados para a sinalização WebRTC**. Esta é a configuração que define quais os portos que podem ser utilizados pelas OneMeetingRooms para a sinalização WebRTC. Ao contrário da sinalização SIP, a sinalização WebRTC só é suportada através da comunicação em TCP.

- **Usar IP Externo para Sinalização WebRTC**. Esta é uma configuração de valor booleano semelhante à configuração SIP. Pelos mesmos motivos que a configuração SIP, também esta deve ser configurada com o valor "*false*"devendo o IP escolhido ser acessível pelos OneMeetingWebRTCServices que servem de *proxy* aos participantes WebRTC.

- **Portos de media SIP**. Cada sala necessita de uma coleção de portos de *media* SIP. Esta configuração é exclusiva para cada sala para garantir o bom funcionamento do sistema. A coleção de portos, aqui configurada, deve ser suficientemente abrangente para garantir que existe portos de *media* para todos os participantes em todas as salas.

- **Portos de media WebRTC**. Esta configuração é semelhante à anterior mas para os portos de *media* WebRTC.

4.4 Rest API

As Rest API são APIs criadas como o propósito de melhorar a interoperabilidade do produto e de disponibilizar algumas informações relevantes do sistema. O OneContactMeetingServer, como referido anteriormente, é um produto que adiciona funcionalidades a utilizadores de outros produtos, sendo imperativo a disponibilização de interfaces fáceis de utilizar que permitam a configuração e recolha da informação por parte desses produtos ou a novas aplicações que possam surgir para gerir e interagir com o OneContactMeetingServer. Foi, então, criada três APIs destinadas para servir diferentes propósitos.

4.4.1 API de Provisionamento

A API de provisionamento vai permitir que sejam provisionadas salas de conferências. As ações principais desta API irão ser utilizadas por aplicações externas a este produto com o objetivo de parametrizar as salas de conferências que poderão ser iniciadas. Os parâmetros principais no provisionamento das salas são: o ID do dono da sala, o número máximo de participantes, se dá suporte a vídeo, se dá suporte a participantes SIP, entre outras.

4.4.2 Runtime API

Esta API disponibiliza métodos para pesquisar as salas que estão em execução atualmente, as OneMeetingPools disponíveis e a carga de cada OneMeetingPool. Estes métodos são úteis para o desenvolvimento de aplicações de monitorização do sistema.

Esta API poderá também ser utilizada por outros componentes no momento de criação de salas ou no momento em que se juntam participantes a salas de conferência em execução.

4.4.3 API de histórico

Esta API tem a função fornecer dados históricos sobre o OneContactMeetingServer. Esta API tem como objetivo fornecer dados históricos sobre o sistema. Estes dados podem ser tanto das salas que executaram como dos eventos que aconteceram no sistema, como falhas ou indisponibilidade de componentes.

4.5 Comunicação

Nesta secção irão ser apresentados os diagramas de comunicação entre os módulos anteriormente apresentados.

Os diagramas pretendem demonstrar o fluxo de comunicação nos quatro cenários principais. Destes, dois são relativos à inicialização de uma sala de conferências pelo um participante SIP e WebRTC e os outros dois são referentes ao processo de adição de um novo participante SIP e WebRTC quando a conferência já se encontra a decorrer.

4.5.1 Inicialização de sala por um participante WebRTC

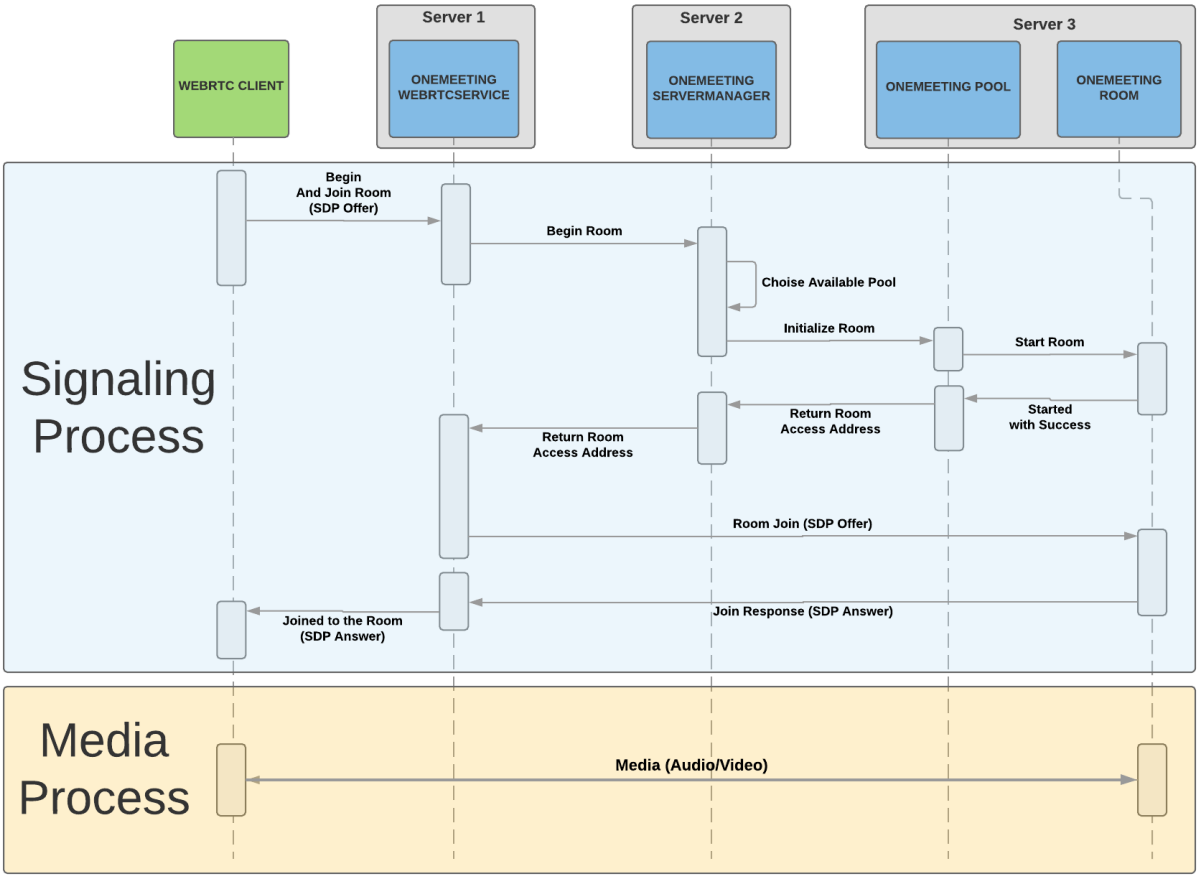

Figura 19: Inicialização de sala por um participante WebRTC

Na Figura 19 está apresento o fluxo de comunicação de um participante WebRTC a iniciar uma sala de conferência. O diagrama é explicado nos seguintes passos:

1. O cliente envia um pedido para o OneMeetingWebRTCService, que serve de *proxy* aos participantes WebRTC. Este pedido contém, entre outras coisas, a identificação do utilizador e o SDP, com descrição necessária à negociação dos canais de media.

2. O OneMeetingWebRTCService faz o pedido de inicialização ao OneMeetingServerManager, o componente que faz a gestão de todo o sistema. Neste pedido o SDP não é enviado pois este não necessita de ter a descrição dos *medias*.

3. Após algumas validações sobre o utilizador, que não estão apresentadas no diagrama, o OneMeetingServerManager elege uma OneMeetingPool para receber a conferência e envia o pedido para iniciar a sala

4. A OneMeetingPool inicia uma OneMeetingRoom para suportar a conferência e aguarda pela confirmação da OneMeetingRoom.

5. Após a confirmação da OneMeetingRoom, a OneMeetingPool responde ao OneMeetingServerManager com a informação de acesso à OneMeetingRoom. Esta informação contém o endereço de IP e portos para acesso dos participantes SIP e WebRTC.

6. O OneMeetingServerManager guarda a informação proveniente da OneMeetingPool e responde ao OneMeetingWebRTCServices com a mesma informação.

7. O OneMeetingWebRTCServices lê o endereço de IP e porto para o *endpoint* WebRTC e abre uma conexão direta à sala. Após a conexão estabelecida, encaminha a mensagem que recebeu do cliente para a OneMeetingRoom.

8. A OneMeetingRoom processa o pedido e envia a resposta com o seu SDP, aceitando assim o pedido de início da sessão.

9. O OneMeetingRoomWebRTCService encaminha a resposta para o cliente.

10. O cliente e a OneMeetingRoom negoceiam diretamente os canais de media e começam a trocar áudio e vídeo.

4.5.2 Novo participante WebRTC

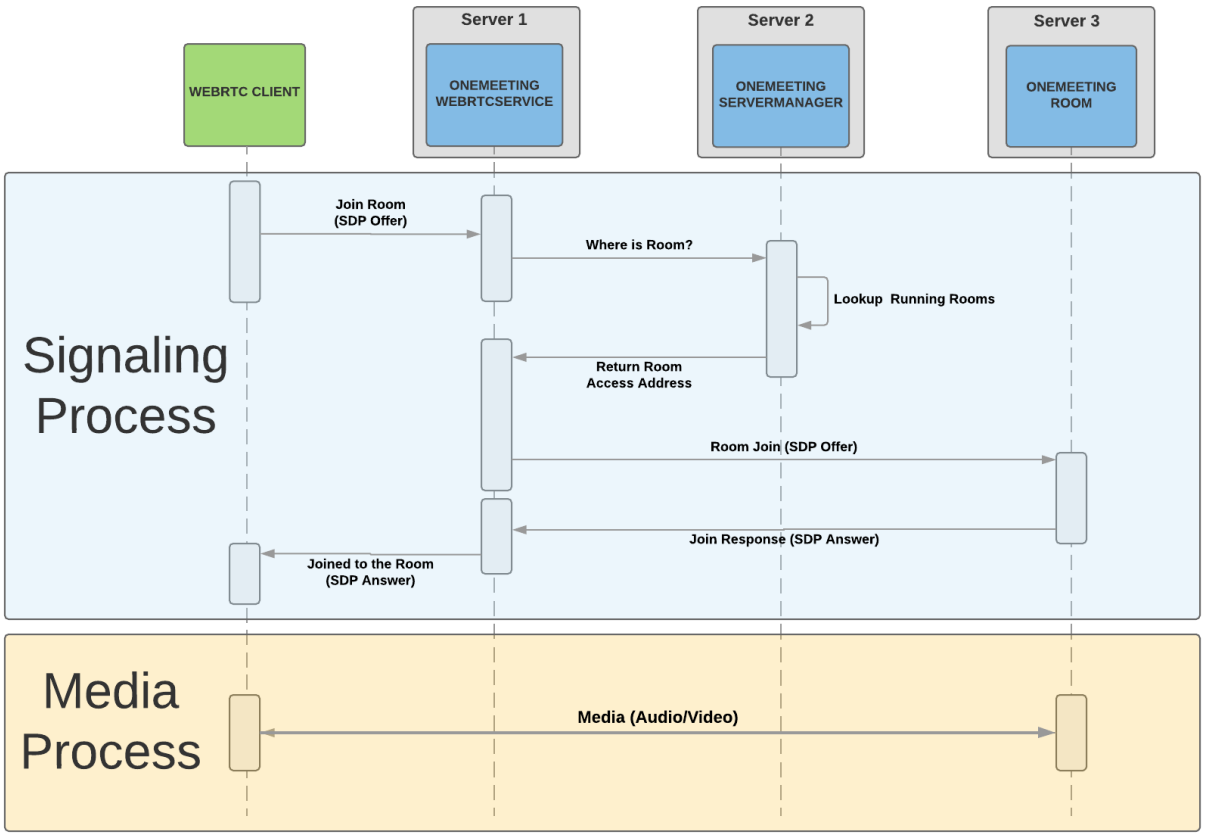

Figura 20: Novo participante WebRTC

Na Figura 20 está apresento o fluxo de comunicação quando um participante WebRTC a entrar numa sala previamente inicializada. O fluxo é semelhante ao anterior, com a diferença que o OneMeetingServerManager não necessita de comunicar com a OneMeetingPool. Os passos são os seguintes:

1. O cliente envia um pedido ao OneMeetingWebRTCService com o identificador da conferência que pretende entrar. À semelhança do diagrama anterior, este pedido contém também o SDP do cliente.

2. O OneMeetingWebRTCService faz o pedido de inicialização ao OneMeetingServerManager a informar que existe um participante que pretende entrar na conferência.

3. O OneMeetingServerManager procura a conferência pretendia e responde com o endereço IP e porto da OneMeetingRoom onde decorre a conferência.

4. O OneMeetingWebRTCServices lê o endereço de IP e porto para o *endpoint* WebRTC e abre conexão direta à sala. Após a conexão estabelecida, encaminha a mensagem que recebeu do cliente para a OneMeetingRoom.

5. A OneMeetingRoom processa o pedido e envia a resposta com o seu SDP, aceitando assim o pedido de início da sessão.

6. O OneMeetingRoomWebRTCService encaminha a resposta para o cliente.

7. O cliente e a OneMeetingRoom negoceiam diretamente os canais de media e começam a trocar áudio e vídeo.

4.5.3 Inicialização de sala por um participante SIP

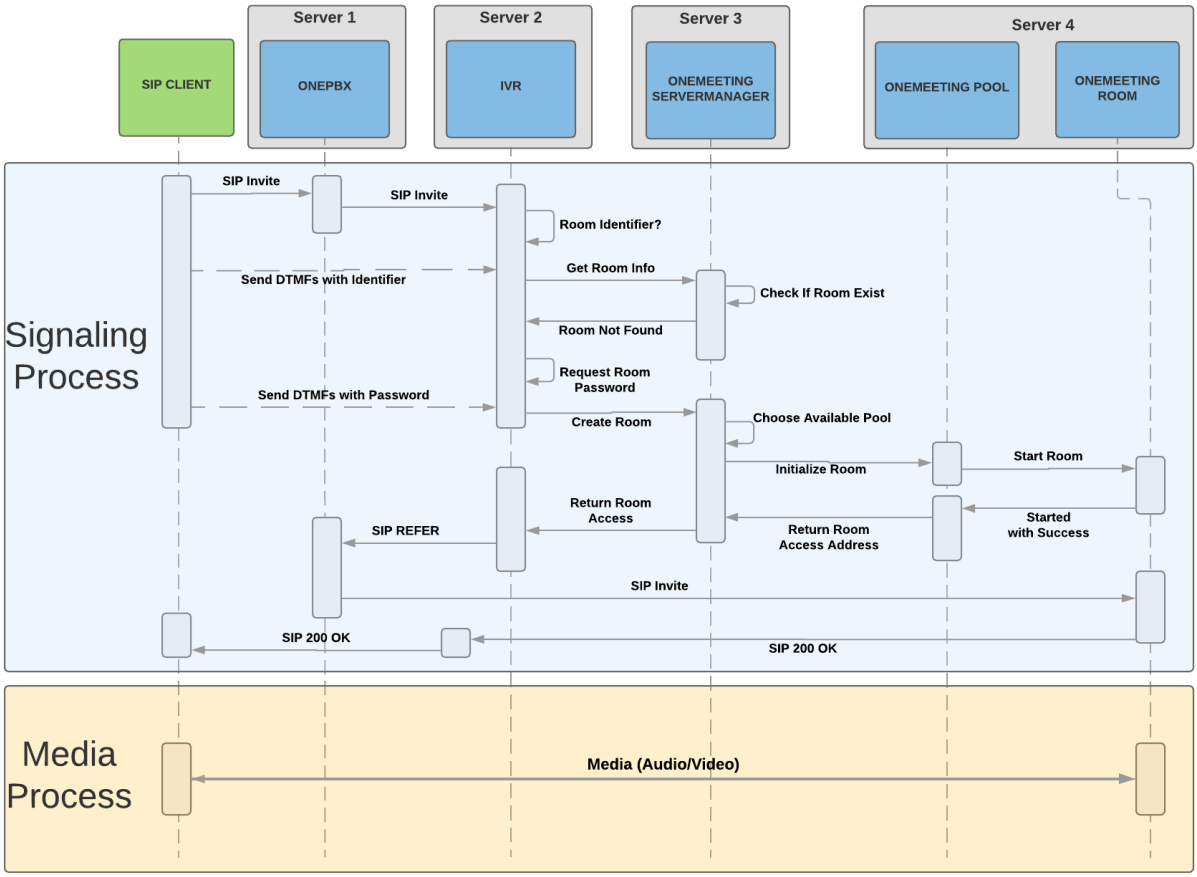

Figura 21: Inicialização de sala por um participante SIP

O OneContactMeetingServer suporta dois tipos diferentes de salas e cada uma pode endereçar várias necessidades diferente, como explicado na secção 3.4. A variedade dos cenários suportados causa algumas diferenças nos fluxos de comunicação dos participantes SIP e dependem da configuração e arquitetura de todo o sistema, incluindo o OneContactPBX. Na Figura 21 está apresentado o cenário referente às *Owned Meeting Rooms* em que os participantes necessitam de fazer uma chamada para um dado DDI que é atendido por um IVR. Só as mensagens mais relevantes trocadas entre os componentes é que estão apresentadas no diagrama tendo em vista a simplificação do fluxo para facilitar a interpretação. A comunicação apresentadas na Figura 21 segue a seguinte sequência:

1. O cliente inicia uma sessão para para o IVR, telefonado para um número específico (DDI).

2. O IVR atende a chamada e pede o identificador da sala de conferência ao utilizador.

3. O utilizador digita o identificador da sala de conferência.

4. O IVR faz um pedido ao OneMeetingServerManager a pedir as informações. O pedido, entre outras coisas, contém o identificador da sala.

5. O OneMeetingServerManager responde que a sala pretendida não foi encontrada e por isso não está em execução.

6. O IVR passa a informação ao utilizador e pede para digitar a *password* caso seja o dono da sala.

7. O utilizador digita a *password*.

8. O IVR faz novo pedido ao OneMeetingServerManager a pedir a informação da sala. O pedido contém, entre outras coisas, o identificador e a *password*.

9. O OneMeetingServerManager volta a procurar a sala pretendida e se não encontrar, este elege um OneMeetingPool disponível e envia o pedido para iniciar a sala.

10. A OneMeetingPool inicia uma OneMeetingRoom para suportar a conferência e aguarda a confirmação da OneMeetingRoom.

11. Após a confirmação da OneMeetingRoom, a OneMeetingPool responde ao OneMeetingServerManager com a informação de acesso à OneMeetingRoom.

12. O OneMeetingServerManager guarda a informação proveniente da OneMeetingPool e responde ao IVR com a informação de acesso.

13. O IVR transfere a chamada para a OneMeetingPool, enviando a mensagem SIP "REFER".

14. O OnePBX envia o pedido de início de sessão para a OneMeetingRoom.

15. A OneMeetingRoom responde com o código 200(OK) para o OnePBX.

16. O OnePBX encaminha a resposta para o cliente.

17. O cliente e a OneMeetingRoom começam a trocar diretamente os pacotes de *media*.

4.5.4 Novo participante SIP

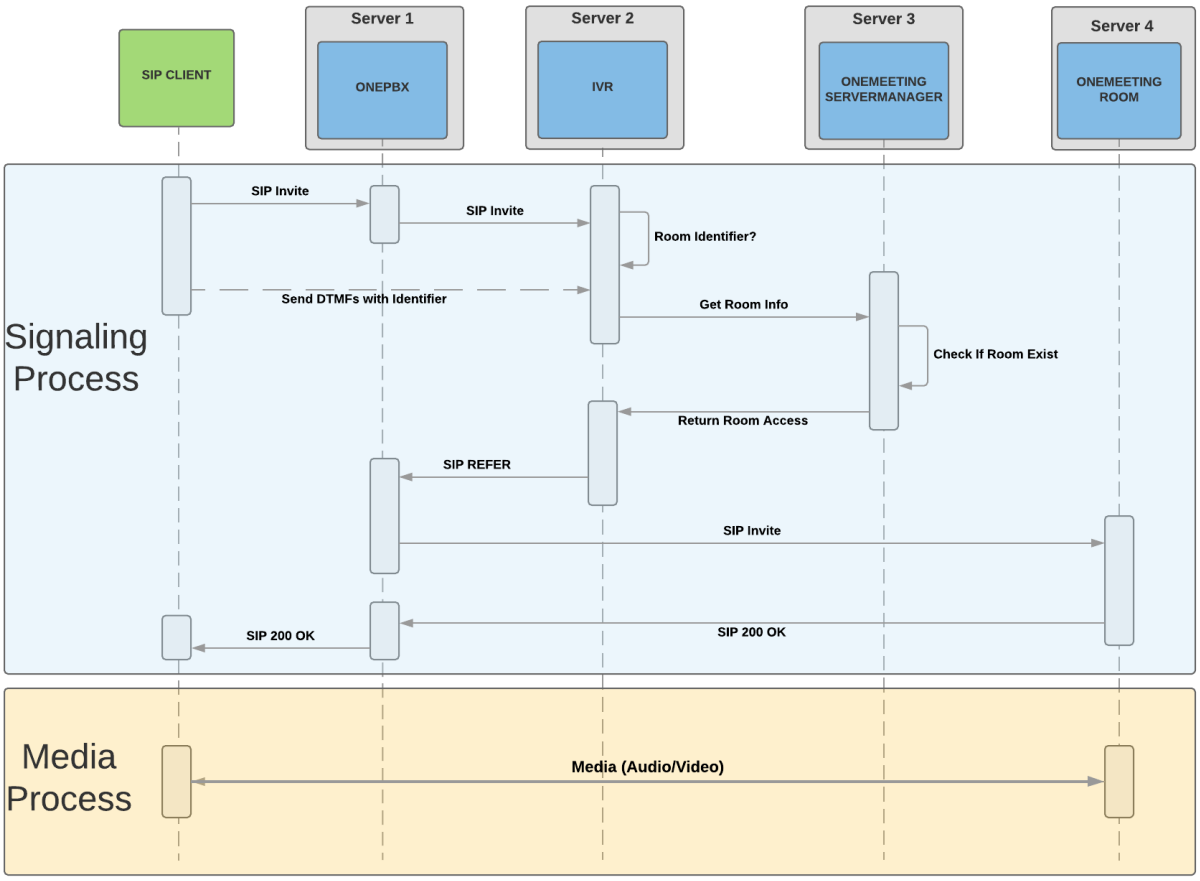

Figura 22: Novo participante SIP

Como foi referido anteriormente, existem vários tipos de salas e os fluxos de comunicação SIP podem ser diferentes. O fluxo apresentado na Figura 22 é o fluxo de um participante a junta-se a una sala de conferência previamente criada segundo fluxo da Figura 21.A comunicação apresentada segue a seguinte sequência:

1. O cliente inicia uma sessão para para o IVR, telefonado para um número específico (DDI).

2. O IVR atende a chamada e pede o identificador da sala de conferência ao utilizador.

3. O utilizador digita o identificador da sala de conferência.

4. O IVR faz um pedido ao OneMeetingServerManager a pedir as informações. O pedido, entre outras coisas, contém o identificador da sala.

5. O OneMeetingServerManager procura a sala pretendida e responde com a informação de acesso.
6. O IVR transfere a chamada para a OneMeetingPool, enviando a mensagem SIP "REFER".
7. O OnePBX envia o pedido de início de sessão para a OneMeetingRoom.
8. A OneMeetingRoom responde com o código 200(OK) para o OnePBX.
9. O OnePBX encaminha a resposta para o cliente.
10. O cliente e a OneMeetingRoom começam a trocar diretamente os pacotes de *media*.

4.6 Síntese

Este capítulo apresentou em detalha cada um dos componentes do OneContactMeetingServer. Demonstrou o processo de sinalização das salas, a negociação do canais de media, o funcionamento do *Audio Mixer* e como as streams de vídeo são geridas. Tentou também demonstrar como a gestão do sistema é feita, a forma como a carga é distribuída e quais as interfaces disponibilizadas aos componentes externos. Por fim, tentou também demonstrar quais as configurações necessárias para por toda a solução em funcionamento.

5. Resultados

Este capítulo destina-se à apresentação dos resultados obtidos no final deste projeto. Em seguida é apresentado alguns cenários testados com o intuito de exibir o funcionamento do sistema.

Como foi referido anteriormente, o OneMeetingServerManager é o componente que faz a gestão de todo o sistema. Este recebe o registo das OneMeetingPool e decide onde uma conferência executa. Os teste foram executados com duas OneMeetingPools registadas no sistema e com um OneMeetingServerManager.

Os resultados apresentados focam-se na demonstração do servidor desenvolvido. Não serão demonstrados resultados sobre aplicações cliente, pois estas não estão incluídas no âmbito deste projeto.

5.1 Registo das OneMeetingPools

Este caso de teste pretende demonstrar o registo de duas OneMeetingPools no OneMeetingServerManager para que estas possam ficar elegíveis para receber salas de conferência. O plano de execução deste teste obedece à seguinte sequências de passos:

1. Iniciar o serviço do OneMeetingServerManager.
2. Iniciar o serviço da OneMeetingPool 1 presente na máquina com o IP 10.0.0.12.
3. Iniciar o serviço da OneMeetingPool 2 presente na máquina com o IP 10.0.0.11.

O objetivo do teste é validar que as duas OneMeetingPools ficam registadas no sistema e disponíveis para receber salas de conferência.

```
1  22:24:11.297     1  INFO Log File: C:\Collab\Logs\OneMeetinngServer\OneMeetingPool.log
2  22:24:11.531     1  INFO Log Level: Info
3  22:24:11.531     1  INFO Log Async: True
4  22:24:11.719     1  INFO PoolId: 01c2cc5c-065e-4401-9506-706083852e71
5  22:24:11.766     5  INFO Room process path: C:\Program Files\Collab\OneMeetingPool\OneMeetingRoom.exe
6  22:24:11.859     5  INFO On first connect to OneMeetingServerManager Event
```

Figura 23: OneMeetingPool 1 no cenário de teste 1

```
1  22:24:22.642     1  INFO Log File: C:\Collab\Logs\OneMeetinngServer\OneMeetingPool.log
2  22:24:22.782     1  INFO Log Level: Info
3  22:24:22.782     1  INFO Log Async: True
4  22:24:23.001     1  INFO PoolId: 20612a3b-6390-4aac-82a4-a35f11f5767d
5  22:24:23.032     5  INFO Room process path: C:\Program Files\Collab\OneMeetingPool\OneMeetingRoom.exe
6  22:24:23.190     5  INFO On first connect to OneMeetingServerManager Event
```

Figura 24: OneMeetingPool 2 no cenário de teste 1

```
1  22:23:59.459     1  INFO Log File: C:\Collab\Logs\OneMeetinngServer\OneMeetingServerManager.log
2  22:23:59.640     1  INFO Log Level: Info
3  22:23:59.643     1  INFO Log Async: True
4  22:24:12.148     9  INFO Pool 01c2cc5c-065e-4401-9506-706083852e71 registered --> 10.0.0.12:49359
5  22:24:23.183     9  INFO Pool 20612a3b-6390-4aac-82a4-a35f11f5767d registered --> 10.0.0.11:49286
```

Figura 25: OneMeetingServerManager no cenário de teste 1

A Figura 23 e a Figura 24 mostram o início das OneMeetingPools e o seu registo

- Linha 4 da Figura 23 "22:24:11.719 INFO PoolId: 01c2cc5c-065e-4401-9506-706083852e71"- Esta linha mostra que a OneMeetingPool 1 iniciou com o identificador "01c2cc5c-065e-4401-9506-706083852e71".
- Linha 6 Figura 23 "22:24:11.859 INFO On first connect to OneMeetingServerManager Event"- Aqui está representado o evento despoletado no final da negociação do *Socket* TCP da OneMeetingPool 1 com o OneMeetingServerManager.
- Linha 4 Figura 24 "22:24:23.001 INFO PoolId: 20612a3b-6390-4aac-82a4-a35f11f5767d"- Esta linha mostra que a OneMeetingPool 2 iniciou com o identificador "20612a3b-6390-4aac-82a4-a35f11f5767d".
- Linha 6 Figura 24 "22:24:23.190 INFO On first connect to OneMeetingServerManager Event"- Aqui está representado o evento despoletado no final da negociação do *Socket* TCP da OneMeetingPool 2 com o OneMeetingServerManager.
- Linha 4 da Figura 25 "22:24:12.148 INFO Pool 01c2cc5c-065e-4401-9506-706083852e71 registered –> 10.0.0.12:49359"- Pode-se constatar nesta linha que a a OneMeetingPool 1 ficou registada no OneMeetingServerManager com o IP 10.0.0.12 e o porto de saída número 49359.

- Linha 5 da Figura 25 "22:24:23.183 INFO Pool 20612a3b-6390-4aac-82a4-a35f11f5767d registered –> 10.0.0.11:49286"- Pode-se constatar nesta linha que a a OneMeetingPool 2 ficou registada no OneMeetingServerManager com o IP 10.0.0.11 e o porto de saída número 49286.

5.2 Início de duas salas de conferência

Este caso de teste pretende demonstrar a inicialização de duas *Instant Meeting Rooms*, explicadas anteriormente na secção 3.4, por participantes SIP. Neste cenário os participantes fazem uma chamada para um DDI que é atendido por um IVR. O IVR pede o identificador da sala ao participante e faz um pedido ao OneMeetingServerManager com o objetivo de saber para onde deve transferir a chamada. O plano de execução deste teste obedece à seguinte sequência de passos:

1. O participante 1 cria uma sala com o identificador "123456".

2. O participante 2 junta-se a sala criada pelo participante 2.

3. O participante 3 cria uma sala com o identificador "654321".

4. O participante 4 junta-se à sala criada pelo participante 2.

Com este teste é esperado que sejam criadas duas salas de conferência em OneMeetingPools diferentes e que os participantes 2 e 4 se consigam juntar às respetivas salas de conferência.

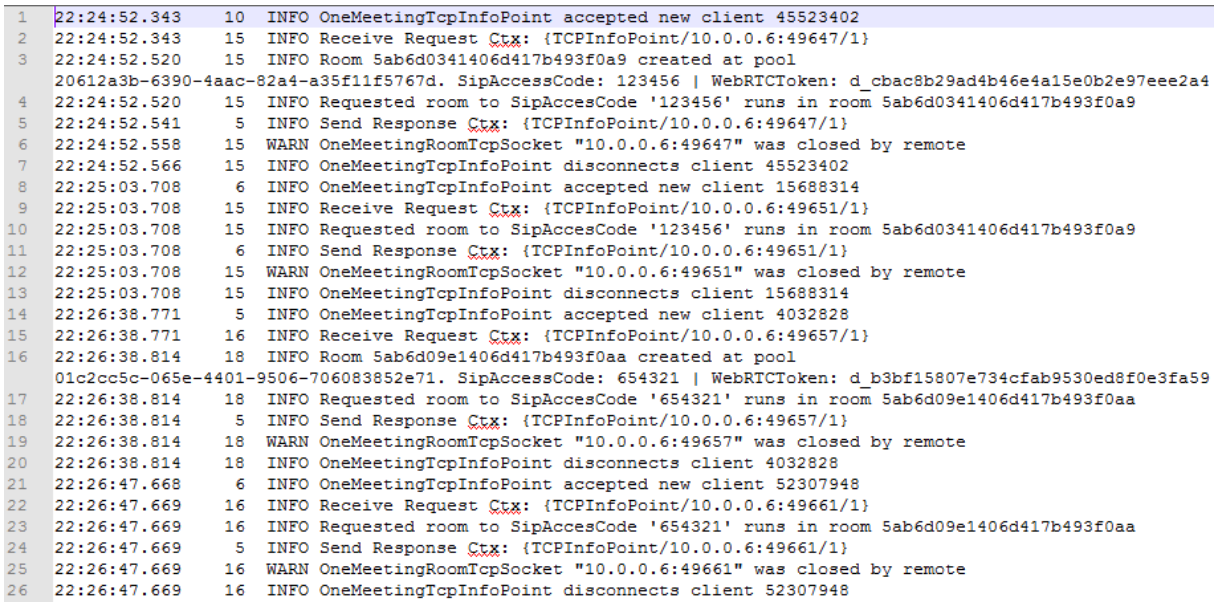

Figura 26: OneMeetingServerManager no caso de teste 2

A Figura 26 pode ser interpretada da seguinte forma:

- Linha 4 "22:24:52.343 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.6:49647/1"- Esta linha demonstra que chegou um pedido do IVR ao OneMeetingServerManager.
- Linha 6 "22:24:52.520 INFO Room 5ab6d0341406d417b493f0a9 created at pool 20612a3b-6390-4aac-82a4-a35f11f5767d. SipAccessCode: 123456 WebRTCToken: d_cbac8b29ad4b46e4a15e0b2e97eee2a4 - Esta linha demonstra que foi criada uma sala de conferência com o identificador SIP "123456"na OneMeetingPool 2 .
- Linha 9 "22:25:03.708 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.6:49651/1"- Esta linha mostra que recebeu um novo pedido do IVR.
- Linha 10 "22:25:03.708 INFO Requested room to SipAccesCode '123456' runs in room 5ab6d0341406d417b493f0a9"- Esta linha demonstra que o segundo pedido é para a sala com o identificador "123456"e que a sala foi encontrada. Este pedido corresponde ao participante 2 para se juntar a sala criada pelo participante 1.
- Linha 15 "22:26:38.771 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.6:49657/1"- Esta linha demonstra que o OneMeetingServerManager recebe outro pedido do IVR, correspondente ao participante 3.
- Linha 16 "22:26:38.814 INFO Room 5ab6d09e1406d417b493f0aa created at pool 01c2cc5c-065e-4401-9506-706083852e71. SipAccessCode: 654321 | WebRTCToken: d_b3bf15807e734cfab9530ed8f0e3fa59 - Aqui pode-se constatar que foi criada uma nova sala para o identificador SIP "654321"na OneMeetingPool 1.

- Linha 22 "22:26:47.669 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.6:49661/1"- Aqui está representado a receção de um novo pedido do IVR.
- Linha 23 "22:26:47.669 INFO Requested room to SipAccesCode '654321' runs in room 5ab6d09e1406d417b493f0aa"- Esta linha demonstra que o pedido da linha anterior destina-se à sala com o identificador "654321"e que a sala foi encontrada.

```
1 22:24:52.461    14  INFO Room 1 state change from ROOM_PROCESS_STATE_READY to
  :ROOM_PROCESS_STATE_WAIT_FOR_RUNNING_CONFIRMATION
2 22:24:52.476    14  INFO Room 3 state change from ROOM_PROCESS_STATE_NOT_STARTED to
  :ROOM_PROCESS_STATE_STARTING
3 22:24:52.476    14  INFO Room 1 state change from ROOM_PROCESS_STATE_WAIT_FOR_RUNNING_CONFIRMATION to
  :ROOM_PROCESS_STATE_RUNNING
4 22:24:52.529    17  INFO Room 3 state change from ROOM_PROCESS_STATE_STARTING to
  :ROOM_PROCESS_STATE_SATRTED_WAIT_FOR_READY
5 22:24:52.545    17  INFO Room 3 state change from ROOM_PROCESS_STATE_SATRTED_WAIT_FOR_READY to
  :ROOM_PROCESS_STATE_READY
6 22:24:52.545    17  INFO Room 3 has been added to free rooms queue
```

Figura 27: OneMeetingPool 2 no caso de teste 2

Na Figura 27 é mostrado os eventos da OneMeetingPool 2 quando a primeira sala de conferência é criada. As linhas relevantes são:

- Linha 1 "22:26:38.849 INFO Room 1 state change from ROOM_PROCESS_STATE_READY"to :ROOM _PROCESS_STATE_WAIT_FOR_RUNNING_CONFIRMATION - Esta linha demonstra que a OneMeetingPool troca de estado uma OneMeetingRoom previamente criada e que se encontrava livre. Neste momento ficou à aguardar a confirmação da OneMeetingRoom.

- Linha 3 "22:26:38.849 INFO Room 1 state change from ROOM_PROCESS_STATE_WAIT_FOR _RUNNING_CONFIRMATION to :ROOM_PROCESS_STATE_RUNNING"- Esta linha demonstra que a OneMeetingPool troca o estado da OneMeetingRoom para o estado de execução. Esta troca de estado acontece quando a OneMeetingRoom confirma à OneMeetingPool que já se encontra em execução. É neste momento que a OneMeetingPool responde ao OneMeetingServerManager com a informação sobre a OneMeetingRoom que suporta a conferência pedida.

- Linha 2 "22:26:38.849 INFO Room 3 state change from ROOM_PROCESS_STATE_NOT_STARTED to :ROOM _PROCESS_STATE_STARTING"- Esta linha demonstra que a OneMeetingPool inicia um novo processo da OneMeetingRoom. O objetivo é manter sempre duas OneMeetingRooms criadas, livres e disponíveis para receber conferência a qualquer momento para que as respostas ao OneMeetingServerManager sejam feitas o mais rápido possível.

- Linha 6 "22:26:38.908 INFO Room 3 has been added to free rooms queue"- Esta linha mostra que adicionou a nova OneMeetingRoom à fila de salas livres.

```
1  22:24:23.337  7412  I Pipe Name: 4214f24d-1f2e-4958-8335-8a8d90b3f6ae
2  22:24:23.337  7412  I SIP Domain: 10.0.0.11
3  22:24:23.337  7412  I Media Bridge: 52.169.31.206
4  22:24:23.337  7412  I SIP TCP: 10000 SIP UDP:10000 WebRTC:10100
5  22:24:23.337  7412  I SIP Media Ports: [13000-13031]
6  22:24:23.337  7412  I WebRTC Media Ports: [12000-12031]
7  22:24:23.337  7412  I Create named pipe communcicator
8  22:24:23.378  7412  I SIP TCP listener ready for incoming connections at 0.0.0.0:10100
9  22:24:23.382  7412  I Changed room state form: ONE_MEETING_ROOM_STATE_INITIALIZING to ONE_MEETING_ROOM_STATE_I
10 22:24:52.481  8112  I Try change room state form: ONE_MEETING_ROOM_STATE_READY to
   ONE_MEETING_ROOM_STATE_RUNNING_WAITING_FOR_FIRST_PARTICIPANT. Success: true
11 22:24:52.747  8088  I RX 850 bytes Request msg INVITE/cseq=1 (rdata0000007086A04D78) from UDP 10.0.0.5:5060
12
13 22:24:52.747  8088  I INVITE sip:123456@10.0.0.11:10000;pbxop=bxfer SIP/2.0
14 From: "+3516000"<sip:+3516000@10.0.0.5>;tag=1c92d988-500000a-13c4-50030-ffe-657a36a6-ffe
15 To: <sip:123456@10.0.0.11:10000;pbxop=bxfer>
16 Call-ID: 1d2292c0-500000a-13c4-50030-ffe-c799180-ffe
17 CSeq: 1 INVITE
18 Via: SIP/2.0/UDP 10.0.0.5:5060;branch=z9hG4bK-ffe-3e78a7-1dda3f37
19 Max-Forwards: 70
20 Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
21 Contact: <sip:+3516000@10.0.0.5:5060;transport=UDP>
22 Content-Type: application/sdp
23 Content-Length: 314
24
25 v=0
26 o=OneSIPConnector 76 2 IN IP4 10.0.0.11
27 s=OneSIPConnector
28 c=IN IP4 10.0.0.11
29 t=0 0
30 m=audio 30092 RTP/AVP 0 8 18 9 101
31 a=rtpmap:0 PCMU/8000
32 a=rtpmap:8 PCMA/8000
33 a=rtpmap:18 G729/8000
34 a=fmtp:18 annexb=no
35 a=rtpmap:9 G722/8000
36 a=ptime:20
37 a=sendrecv
38 a=rtpmap:101 telephone-event/8000
39 a=fmtp:101 0-15
40
41 22:25:04.043  8096  I RX 905 bytes Request msg INVITE/cseq=1 (rdata0000007086A04D78) from UDP 10.0.0.5:5060
42
43 22:25:04.044  8096  I INVITE sip:123456@10.0.0.11:10000;pbxop=bxfer SIP/2.0
44 From: "+3516000"<sip:+3516000@10.0.0.5>;tag=1c92f368-500000a-13c4-50030-1009-34395e4c-1009
45 To: <sip:123456@10.0.0.11:10000;pbxop=bxfer>
46 Call-ID: 1d229f60-500000a-13c4-50030-1009-662641ce-1009
47 CSeq: 1 INVITE
48 Via: SIP/2.0/UDP 10.0.0.5:5060;branch=z9hG4bK-1009-3ea4c7-12d38dae
49 Max-Forwards: 70
50 Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
51 Contact: <sip:+3516000@10.0.0.5:5060;transport=UDP>
52 Content-Type: application/sdp
53 Content-Length: 363
54
55 v=0
56 o=OneSIPConnector 80 2 IN IP4 10.0.0.11
57 s=OneSIPConnector
58 c=IN IP4 10.0.0.11
59 t=0 0
60 m=audio 31476 RTP/AVP 18 97 101
61 a=alt:1 1 : E9hWfTu3 Fyyr6zoI 172.18.190.117 46158
62 a=fmtp:18 annexb=yes
63 a=fmtp:101 0-15
64 a=rtpmap:18 G729/8000
65 a=rtpmap:97 SPEEX/8000
66 a=rtpmap:101 telephone-event/8000
67 a=sendrecv
68 a=x-rtp-session-id:B83C426A517842C3A29E8D9D6857ACC0
```

Figura 28: OneMeetingRoom no caso de teste 2

Na Figura 28 estão apresentados alguns eventos da OneMeetingRoom que suporta a primeira conferência criada com o identificador SIP "123456". As linha relevantes são:

- Linha 10 "22:24:52.481 I Try change room state form: ONE_MEETING_ROOM_STATE_READY to ONE_MEETING_ROOM_STATE_RUNNING_WAITING_FOR_FIRST_PARTICIPANT. Success: true"-

Esta linha demonstra o momento em que a OneMeetingRoom recebe o pedido para ir para o estado de execução feito pela OneMeetingPool.

- Linha 13 "22:24:52.747 I INVITE sip:123456@10.0.0.11:10000;pbxop=bxfer SIP/2.0"- Momento em que a OneMeetingRoom recebe o participante 1 através da mensagem SIP "INVITE".

- Linha 43 "22:25:04.044 I INVITE sip:123456@10.0.0.11:10000;pbxop=bxfer SIP/2.0"- Momento em que a OneMeetingRoom recebe o participante 2 através da mensagem SIP "INVITE".


```
{
    "_id" : ObjectId("5ab6d0341406d417b493f0a9"),
    "Owner" : "",
    "Pool" : "20612a3b-6390-4aac-82a4-a35f11f5767d",
    "CanDoVideo" : false,
    "CanDoSip" : true,
    "MaxParticpants" : 32,
    "Type" : "instant",
    "State" : "running",
    "SipCode" : "123456",
    "PropertyWebRTCEndpoints" : [
        {
            "Endpoint" : {
                "Host" : "52.169.31.206",
                "Port" : 10100
            },
            "Proto" : 2
        }
    ],
    "PropertySipEndpoints" : [
        {
            "Endpoint" : {
                "Host" : "10.0.0.11",
                "Port" : 10000
            },
            "Proto" : 2
        },
        {
            "Endpoint" : {
                "Host" : "10.0.0.11",
                "Port" : 10000
            },
            "Proto" : 1
        }
    ],
    "AccessToken" : "d_cbac8b29ad4b46e4a15e0b2e97eee
    "V" : 1,
    "lastModified" : {
        "$date" : 1521930292463
    },
    "dateAdded" : {
        "$date" : 1521930292463
    }
}
```


(a) Sala 123456


```
{
"_id" : ObjectId("5ab6d09e1406d417b493f0aa"),
"Owner" : "",
"Pool" : "01c2cc5c-065e-4401-9506-706083852e71",
"CanDoVideo" : false,
"CanDoSip" : true,
"MaxParticpants" : 32,
"Type" : "instant",
"State" : "running",
"SipCode" : "654321",
"PropertyWebRTCEndpoints" : [
    {
        "Endpoint" : {
            "Host" : "40.113.13.221",
            "Port" : 10100
        },
        "Proto" : 2
    }
],
"PropertySipEndpoints" : [
    {
        "Endpoint" : {
            "Host" : "10.0.0.12",
            "Port" : 10000
        },
        "Proto" : 2
    },
    {
        "Endpoint" : {
            "Host" : "10.0.0.12",
            "Port" : 10000
        },
        "Proto" : 1
    }
],
"AccessToken" : "d_b3bf15807e734cfab9530ed8f0e3fa59",
"V" : 1,
"lastModified" : {
    "$date" : 1521930398804
},
"dateAdded" : {
    "$date" : 1521930398804
}
}
```


(b) Sala 654321

Figura 29: Documentos na BD das salas criadas no caso de teste 2

Na Figura 29 estão apresentados os registos guardados pelo OneMeetingServerManager na base de dados das duas salas criadas neste caso de teste.

5.3 Início de uma sala conferência com participantes SIP e WebRTC

Este caso de teste pretende demonstrar a inicialização de uma *Owned Meeting Room* por um participante WebRTC e a coexistência de participantes WebRTC e SIP na mesma sala de conferência. Neste cenário o participante WebRTC inicia a sala de conferência e posteriormente dois participantes, um SIP e outro WebRTC, junta-se à conferência. O plano de execução obedece à seguinte sequência de passos:

- O participante 1 é um participante WebRTC e inicia a sua sala de conferência.
- O participante 2 é um participante WebRTC e acede à sala de conferência através do link da sala do Participante 1.
- O participante 3 é um participante SIP e junta-se à sala telefonando para um DDI e digitado o identificador SIP da sala do participante 1.

```
1  22:28:35.229    15  INFO OneMeetingTcpInfoPoint accepted new client 56612102
2  22:28:35.229    17  INFO Receive Request Ctx: {TCPInfoPoint/10.0.0.4:46648/1}
3  22:28:35.298    17  INFO Room 5ab6d0681406d417b493f0ab created at pool
   20612a3b-6390-4aac-82a4-a35f11f5767d. SipAccessCode: 987654 | WebRTCToken:
   AQIAOTkAxG6G6D47oUmSPmDLL4gM9A
4  22:28:35.298    17  INFO Requested room to user id '99' runs in room 5ab6d0681406d417b493f0ab
5  22:28:35.298     5  INFO Send Response Ctx: {TCPInfoPoint/10.0.0.4:46648/1}
6  22:28:35.298    15  WARN OneMeetingRoomTcpSocket "10.0.0.4:46648" was closed by remote
7  22:28:35.566    15  INFO OneMeetingTcpInfoPoint disconnects client 56612102
8  22:28:58.578     6  INFO OneMeetingTcpInfoPoint accepted new client 26598224
9  22:28:58.581    15  INFO Receive Request Ctx: {TCPInfoPoint/10.0.0.4:50389/1}
10 22:28:58.581    15  INFO Requested room to WebRTCToken 'AQIAOTkAxG6G6D47oUmSPmDLL4gM9A' runs in room
   5ab6d0681406d417b493f0ab
11 22:28:58.581     6  INFO Send Response Ctx: {TCPInfoPoint/10.0.0.4:50389/1}
12 22:28:58.581    15  WARN OneMeetingRoomTcpSocket "10.0.0.4:50389" was closed by remote
13 22:28:58.581    15  INFO OneMeetingTcpInfoPoint disconnects client 26598224
14 22:29:16.882     5  INFO OneMeetingTcpInfoPoint accepted new client 5172120
15 22:29:16.882    19  INFO Receive Request Ctx: {TCPInfoPoint/10.0.0.6:50662/1}
16 22:29:16.882    20  INFO Requested room to SipAccesCode '987654' runs in room 5ab6d0681406d417b493f0ab
17 22:29:16.882    20  INFO Send Response Ctx: {TCPInfoPoint/10.0.0.6:50662/1}
18 22:29:16.882    20  WARN OneMeetingRoomTcpSocket "10.0.0.6:50662" was closed by remote
19 22:29:16.882     5  INFO OneMeetingTcpInfoPoint disconnects client 5172120
```

Figura 30: OneMeetingServerManager no caso de teste 3

Na Figura 30 está apresentado o OneMeetingServerManager durante a execução deste cenário de teste. As linhas relevantes são:

- Linha 2 "22:28:35.229 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.4:46648/1"- O OneMeetingServerManager recebe o pedido do primeiro participante vindo do OneMeetingWebRTCServices (a camada de serviços que serve de *proxy* aos participantes WebRTC) que se encontra em execução na máquina com o endereço de IP 10.0.0.4.

- Linha 3 "22:28:35.298 INFO Room 5ab6d0681406d417b493f0ab created at pool 20612a3b-6390-4aac-82a4-a35f11f5767d. SipAccessCode: 987654 | WebRTCToken: AQIAOTkAxG6G6D47oUmSPmDLL4gM9A"- Esta linha indica que a sala do participante 1 foi criada e tem o identificador SIP "987654"e o *token* de acesso aos participantes WebRTC é o "AQIAOTkAxG6G6D47oUmSPmDLL4gM9A".

- Linha 9 "22:28:58.581 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.4:50389/1"- O OneMeeting-ServerManager recebe o pedido do participante 2 vindo do OneMeetingWebRTC-Services.

- Linha 10 "22:28:58.581 INFO Requested room to WebRTCToken 'AQIAOTkAxG6G6D47oUmSPmDLL4gM9A' runs in room 5ab6d0681406d417b493f0ab"- Esta linha indica que o participante 2 pediu a sala de conferência com o *token* "AQIAOTkAxG6G6D47oUmSPmDLL4gM9A"e que a sala se encontra em execução.

- Linha 15 "22:29:16.882 INFO Receive Request Ctx: TCPInfoPoint/10.0.0.6:50662/1"- O OneMeeting-ServerManager recebe o pedido do participante 3 vindo do IVR.

- Linha 16 "22:29:16.882 INFO Requested room to SipAccesCode '987654' runs in room 5ab6d0681406d417b493f0ab"- O participante 3 faz um pedido para o identificador SIP "987654"e a sala já se encontra em execução.

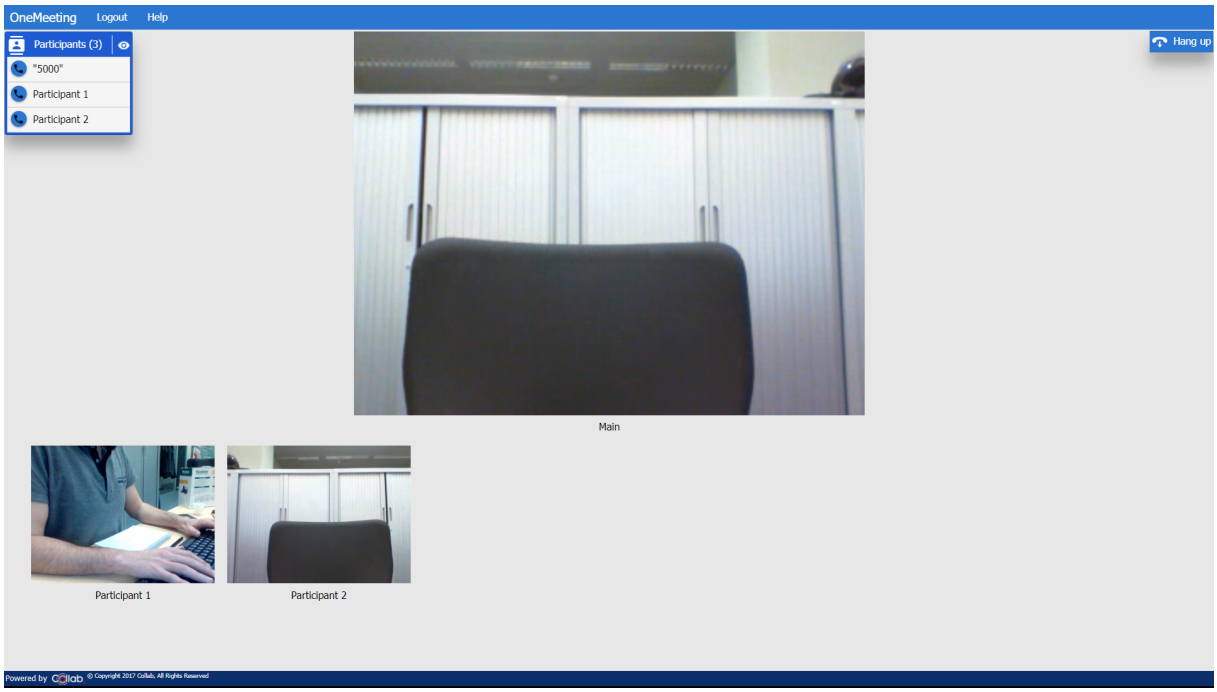

Figura 31: Participante WebRTC

Na Figura 31 está uma aplicação cliente desenvolvida especificamente para se ligar ao OneContactMeetingServer mas que não é objeto de deste projeto. Esta imagem é a

aplicação do participante 1 e onde se pode ver três vídeos diferentes. O vídeo maior é o vídeo principal e que mostra o participante que está a falar em cada momento. Os outros dois são os vídeos secundários que correspondem a cada um dos participantes, incluindo o próprio. No canto superior esquerdo está a lista de todos os participantes onde se pode notar que estão ligados três participantes, dois dos quais são participantes WebRTC e o outro é o participante SIP que só participa com áudio.

5.4 Síntese

Este capítulo expôs três cenários de teste que pretendem demonstrar e comprovar o funcionamento do OneContactMeetingServer. O primeiro cenário comprova a possibilidade de existir múltiplas OneMeetingPools registadas no OneMeetingServerManager proporcionando ao sistema um mecanismo de escalabilidade importante para servir um grande número de utilizadores. O segundo teste demonstra a criação de duas salas e que o OneMeetingServerManager faz a distribuição correta das salas pelas OneMeetingPools disponíveis. O terceiro e último teste pretende comprovar o suporte para a coexistências de participantes SIP e WebRTC na mesma sala de conferência.

6. Conclusão

O objetivo principal deste projeto foi conseguido, o desenvolvimento de um servidor de videoconferência capaz de suportar *endpoints* SIP e WebRTC. O produto desenvolvido responde aos requisitos e necessidades da Collab, criando uma solução *cloud-ready*, escalável e flexível que serve diferentes casos de utilização. Para além do suporte aos *endpoints* SIP e WebRTC era também objetivo a coexistência deste dois tipos de *endpoint* na mesma sala de conferência, objetivo esse que também foi conseguido.

Foi também desenvolvido neste servidor, as funcionalidade que permitem a criação e gestão de salas de conferência de diferente tipos e que servem a diferentes propósitos. Foram também desenvolvidas diferentes formas de interagir com o servidor criando várias APIs e bibliotecas de fácil utilização para que os sistemas externos, como o caso do OneContactPBX, facilmente consigam integrar esta solução.

Ao longo do desenvolvimento foram ultrapassadas várias dificuldades. A maior dificuldade deveu-se à variedade de protocolos necessários para implementação deste servidor. Tanto o protocolo SIP como a tecnologia WebRTC são complexos e fazem uso de muitos outros protocolos auxiliares para conseguirem atingir os objetivos a que se prepõe. A intercomunicação entre estas duas tecnologias exigiu o conhecimento profundo destas e de tudo o que elas envolvem. Também os requisitos não funcionais, destinados a oferecer melhor suporte na *Cloud* no modelo Saas, como por exemplo, a redundância, escalabilidade horizontal, tolerância a falhas, *zero down time* do serviço, entre outras, apresentaram múltiplos desafios na arquitetura do produto e, posteriormente, no seu desenvolvimento.

Outro desafio encontrado foi o facto de o WebRTC ser ainda um *standard* com baixo nível de maturidade que está em constante mudança e que cada *browser* tem diferentes implementações, principalmente no que diz respeito à negociação de várias *streams* de vídeo e áudio.

Apesar das dificuldades apresentadas, no final resulta um produto servidor de videoconferência, o OneContactMeetingServer, capaz de satisfazer todos os requisitos definidos.

Bibliografia

Baugher, M., D. McGrew, M. Naslund, E. Carrara, and K. Norrman (2004, March). The Secure Real-time Transport Protocol (SRTP). RFC 3711, RFC Editor.

Bosse, J. G. V. and F. U. Devetak (2007). *Signaling in telecommunication networks*.

Camarillo, G. and H. Schulzrinne (2010, June). The session description protocol (sdp) grouping framework. RFC 5888, RFC Editor.

Engineering, L. (1997). 5 analog and sampled time domain plot sampled signal frequency plot 5 analog and NRZ sampling time plots 5 analog and sampled signal difference frequency plot for NRZ sampling magifying the frequncy plot X2 oversampling time plot X2 oversampling differenc.

Firestone, S., T. Ramalingam, and S. Fry (2007). *Voice and Video Conferencing Fundamentals*.

Fischl, J., H. Tschofenig, and E. Rescorla (2010, May). Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS). RFC 5763, RFC Editor.

H. Schulzrinne, S. Casner, R. F. and V. Jacobson (2003, July). RTP: A Transport Protocol for Real-Time Applications. RFC 3550, RFC Editor.

Hentschel, T. and G. Fettweis (2001). Sample rate conversion for software radio. *Software Radio Technologies: Selected Readings*, 389–397.

Holmberg, C., H. Alvestrand, and C. Jennings (2017, December). Negotiating media multiplexing using the session description protocol (sdp). Internet-Draft draft-ietf-mmusic-sdp-bundle-negotiation-47, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdp-bundle-negotiation-47.txt`.

Holmberg, C. and R. Shpount (2017, October). Session description protocol (sdp) offer/answer considerations for datagram transport layer security (dtls) and transport layer security (tls). Internet-Draft draft-ietf-mmusic-dtls-sdp-32, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-ietf-mmusic-dtls-sdp-32.txt`.

J. Rosenberg, R. Mahy, P. M. and D. Wing (2008, October). Session Traversal Utilities for NAT (STUN). RFC 5389, RFC Editor.

Johnston, A. B. (2009). *SIP : Understanding the Session Iniation Protocol.*

Johnston, A. B. and D. C. Burnett (2014). *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*, Volume 1.

Lavry, D. (2004). Sampling Theory For Digital Audio. *Lavry Engineering, Inc. Available online: http://www. lavryengineering. com/documents/Sampling_Theory. pdf (checked 24.5. 2010)*, 1–27.

M. Handley, V. J. and C. Perkins (2006, July). SDP: Session Description Protocol. RFC 4566, RFC Editor.

Marquardt, F. (2001). An introduction to the basics of video conferencing. (August), 1–18.

McGrew, D. and E. Rescorla (2010, May). Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764, RFC Editor.

Oracle. JsonRTC protocol reference. `https://docs.oracle.com/cd/E40972_01/doc.70/e40977/xt_wse_protocol_ref.htm`. Accessed: 2018-02-11.

Painter, T. and A. Spanias (2000). Perceptual coding of digital audio. *Proceedings of the IEEE 88*(4), 451–512.

Pan, D. Y. (1993). Digital Audio Compression. *Digital Technical Journal 5*(2), 1–14.

Perkins, C. and M. Westerlund (2010, April). Multiplexing rtp data and control packets on a single port. RFC 5761, RFC Editor.

R. Mahy, P. M. and J. Rosenberg (2010, April). Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766, RFC Editor.

Rosenberg, J. (2010a, April). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, RFC Editor.

Rosenberg, J. (2010b, April). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, RFC Editor.

Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler (2002, June). SIP: Session Initiation Protocol. RFC 3621, RFC Editor.

Sheet, N. A. and V. A. Sheet (2008). Videoconferencing. pp. 1–4.

Uberti, J. (2013, May). Plan b: a proposal for signaling multiple media sources in webrtc. Internet-Draft draft-uberti-rtcweb-plan-00, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-uberti-rtcweb-plan-00.txt`.

Weinstein, I. M. (2012). Real world options for multipoint videoconferencing. (April), 1–11.

Yon, D. and G. Camarillo (2005, September). TCP-Based Media Transport in the Session Description Protocol (SDP). RFC 4145, RFC Editor.