Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado em Engenharia Informática
2º Semestre, 2009/2010

PLAY
Terminal IPTV para Visualização de Sessões de Colaboração Multimédia

Pedro Taborda, 31043

Orientadores
Professor Doutor Pedro Medeiros – FCT/UNL
Engenheiro Paulo Chainho – PT Inovação

28 de Julho de 2010

Nº do aluno: 31043
Nome: Pedro Taborda

Título da dissertação:
PLAY: Terminal IPTV para Visualização de Sessões de Colaboração Multimédia

Palavras-chave:
- IPTV
- televisão
- terminal
- plataforma de comunicação colaborativa

Keywords:
- IPTV
- television
- terminal
- collaborative communication platform

Resumo

Com o aumento da largura de banda disponível e o crescimento do seu alcance geográfico, os operadores de telecomunicações têm neste momento o desafio de desenvolver serviços que acrescentem valor à sua oferta comercial. A crescente integração entre os dispositivos multimédia tradicionais e os serviços de rede abre caminho à produção de soluções que permitam às pessoas tratar esses dispositivos não como componentes individuais mas sim como elementos integrantes de uma plataforma de comunicações multimédia.

Nesse sentido, o objectivo desta tese é o planeamento e implementação de um terminal de colaboração integrado com o serviço de IPTV em casa do utilizador, permitindo que apenas com os habituais dispositivos se possa facilmente estabelecer uma sessão de comunicação colaborativa entre múltiplos participantes.

Foi necessário realizar um levantamento das actuais capacidades dos vários sistemas de IPTV para determinar como proceder à integração de aplicações desenvolvidas por grupos de trabalho externos. Após a análise da plataforma que disponibiliza as funcionalidades colaborativas, procedeu-se à implementação das adaptações necessárias para a adequação às necessidades de um terminal IPTV. Por fim, desenvolveu-se uma aplicação que permite aos utilizadores domésticos desfrutar dessas funcionalidades de comunicação colaborativa através da televisão.

A PT Inovação, tendo uma oferta comercial de IPTV presente no mercado português, poderá de futuro ver a plataforma proposta neste trabalho como um serviço diferenciador que lhe acrescente valor.

Abstract

The recent increase on bandwidth availability and its geographical reach is challenging telecom operators to develop new applications that add value to their commercial offers. Moreover, the extending communication capabilities between both modern and traditional multimedia devices and their collaboration with networked services allows developers to come up with new applications that induce users to acknowledge those devices not as single individual components but as an integrated element of a multimedia platform.

The purpose of this thesis is to plan, develop and implement a colaborative client fully integrated with the user's subscribed IPTV service, allowing to effortlessly establish a multi-featured communication session between multiple participants using the interfaces the user is accustomed to.

A phased plan was established to accomplish the proposed goals. First of all, a survey was conducted on the IPTV service distribution market to analyse each one of these different systems capacities and how they integrate with external applications and devices developed by other people. The following step was the thorough analisys done on the current platform that provides applications with collaborative functionalities, implementing the changes necessary for the integration of an IPTV terminal. Finally, an application was developed to provide a home user with a whole aggregate of colaborative communication services on his TV set.

Having already on the market a fully operational IPTV platform, PT Inovação may find in the future the system proposed in this document to be a differentiative and enriching asset to their offer.

Índice

Índice de figuras

1 Introdução e enquadramento

1.1 Motivação e oportunidade

Com a erosão das receitas dos serviços tradicionais de telecomunicações, os operadores têm feito um forte investimento no lançamento de novas soluções de entretenimento e multimédia, com destaque para os serviços IPTV e Mobile TV. Adicionalmente, a chegada da fibra óptica ao mercado residencial disponibiliza ao cliente larguras de banda superiores a 100Mbps, sendo agora o grande desafio encontrar aplicações que permitam utilizar estes recursos. Neste contexto, a televisão passa a ser encarado como o mais importante terminal no mercado residencial de rede fixa, disponibilizando uma interface unificada e conveniente para o utilizador gerir os seus conteúdos pessoais (filmes, fotografias, música) ou comprados a fornecedores.

Esta aplicação dos conceitos de telepresença no mercado residencial proporciona novas experiências de entretenimento permitindo, por exemplo, reunir um grupo de amigos disperso geograficamente para verem juntos um programa de televisão, um filme alugado, fotografias e filmes gravados nas férias ou para se divertirem com um jogo de rede. Para além destes serviços de lazer, há ainda a possibilidade, de através da televisão e com qualidade de áudio e vídeo de alta definição, disponibilizar novos meios de assistência de saúde, formação, trabalho remoto, entre outros.

Em simultâneo com a mudança de mentalidade dos produtores de serviços, também as empresas que produzem as soluções tecnológicas em que se baseiam estes serviços estão cada vez mais interessadas neste tipo de aplicações. Todos os dias chegam ao mercado doméstico e empresarial novos dispositivos já com suporte aos mais recentes protocolos de comunicação em redes e compatíveis com os mais modernos *codecs* de visualização de áudio e vídeo, que

permitem desenvolver de forma mais simples e com menos custos aplicações que cheguem a todos. Por outro lado, a facilidade de disponibilização destes aparelhos em casa dos clientes aumenta a base de potenciais utilizadores destes novos serviços, o que por sua vez pressiona os produtores de serviços a produzir aplicações que os clientes queiram usar.

Encontramos então neste momento as empresas de telecomunicações num período crucial, em que se cruzam os dois factores mais importantes para o desenvolvimento de produtos de sucesso: a existência de soluções tecnológicas que apoiem os serviços lançados, e o interesse e disponibilidade dos utilizadores em aplicações deste género.

1.2 Problemas a abordar

O desafio proposto no início desta tese de mestrado era, à primeira vista, simples de ultrapassar. Pretendia-se desenvolver uma solução que permitisse utilizar o televisor, historicamente um terminal para mera visualização de conteúdos, de uma forma interactiva como cliente de uma plataforma de serviços colaborativos.

Com o decorrer do tempo e do estudo das tecnologias relacionadas com o trabalho foi-se ganhando a noção de que estes objectivos tinham várias dificuldades associadas. A grande diversidade de soluções que permitem exercer interactividade num televisor e a aparente incompatibilidade entre estas múltiplas soluções obrigou à sua análise crítica de forma a escolher aquelas que por um lado possam chegar de uma forma mais simples e rápida a casa dos utilizadores, e por outro que sejam mais abrangentes geograficamente.

Delineada a plataforma de entrega ao utilizador, foi altura de analisar a fundo plataformas sobre as quais se possam construir serviços de comunicação colaborativa. Neste sentido, procurou-se estudar quais as principais funcionalidades que suportam, quão fácil é a sua utilização por parte de aplicações externas, qual o estado actual do seu desenvolvimento interno e quão difícil se poderá tornar a adição de novas funcionalidades.

Finalmente, procurou-se desenvolver uma solução que, com base no estudo efectuado nos pontos anteriores e nas capacidades das plataformas até então analisadas, permita fornecer ao utilizador as funcionalidades introduzidas no início deste capítulo.

1.3 Contribuições propostas

As contribuições propostas para esta tese de mestrado dividem-se em duas áreas que, embora distintas, se relacionam proximamente. São elas:

- a reformulação e reestruturação do PUC, Plataforma Unificada de Colaboração, solução desenvolvida na PT Inovação que disponibiliza funcionalidades de comunicação colaborativa para diferentes aplicações e em múltiplos terminais.
- a utilização desses serviços colaborativos disponibilizados pelo PUC num terminal IPTV para visualização na televisão. Esta parte do trabalho será daqui para a frente designada PLAY - Terminal IPTV ou simplesmente PLAY.

No início da tese, o PUC encontrava-se ainda em estado *alpha*, muito longe da qualidade necessária para uma solução de produção. Verificou-se que a plataforma sofria de diversos problemas, desde o tempo proibitivo que demorava o processamento das várias operações suportadas, passando pela pouca abstracção da plataforma em relação aos diferentes servidores externos de recursos, e finalizando nos problemas reportados ao nível da API de comunicação entre as aplicações *3rd party* e a plataforma. À vista destes problemas, o objectivo principal desta fase foi a colocação do PUC num estado mais avançado de forma a permitir que outros projectos possam utilizar as suas funcionalidades e serviços de forma mais simples e eficaz.

Paralelamente ao desenvolvimento da nova versão do PUC, e também com base nas conclusões retiradas da sua análise crítica entretanto realizada, pretendeu-se implementar uma solução que possibilite a utilização dos serviços colaborativos disponibilizados pelo PUC na televisão. Neste sentido, o PLAY, terminal de serviços de comunicação colaborativa, deve apresentar-se ao utilizador totalmente integrado com a habitual interface do serviço de IPTV e ser controlado através dos dispositivos a que o utilizador está mais acostumado. A plataforma deve dar aos utilizadores a possibilidade de, na sua própria casa e sem a instalação de equipamento adicional, participarem em sessões de comunicação e colaboração com outros utilizadores, ligados através da televisão ou outros terminais (computador, telefone fixo, *smartphone*). O terminal de televisão deverá permitir a total visualização dos dados

referentes a cada sessão de colaboração (tópico, informação de participantes, informação de recursos, entre outros), e adicionalmente implementar o máximo possível das funcionalidades disponibilizadas pelo PUC (*chat*, videoconferência, *slideshow*, partilha e edição de documentos).

No final do desenvolvimento, devem ser realizados testes de validação a ambas as plataformas que confirmem o seu bom funcionamento e desempenho.

1.4 Organização geral da solução

Como referido na anterior secção, a solução final encontra-se dividida em duas principais plataformas, cada uma com os seus objectivos e áreas de acção. Uma visão é apresentada na Figura 1.

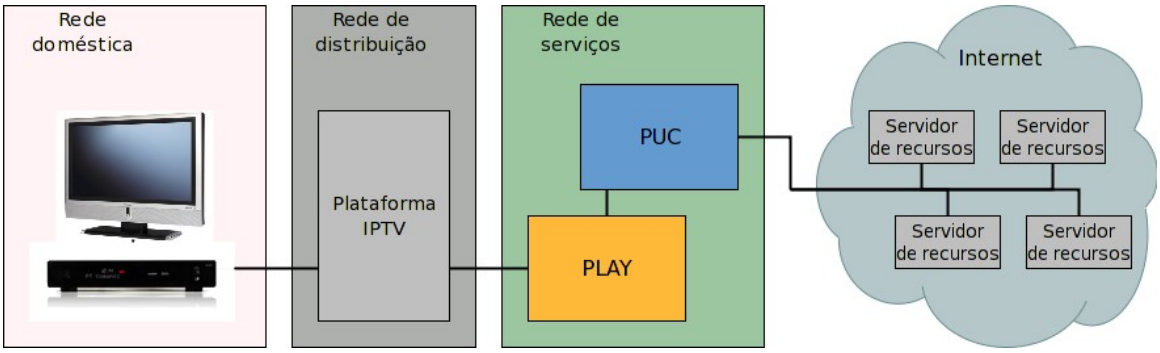

Figura 1: Visão geral da solução

O PUC, plataforma de serviços colaborativos, faz toda a gestão dos dados necessários à criação e manutenção de sessões de colaboração. É da sua responsabilidade controlar a interacção com os vários servidores de recursos externos, implementando uma lógica que controla e fiscaliza o funcionamento destes servidores para que as suas funcionalidades possam ser integradas numa solução unificada.

O PLAY, terminal de serviços colaborativos, é o responsável por toda a comunicação com o serviço de IPTV. É este componente que disponibiliza a interface de utilização através da qual os utilizadores em casa interagem uns com os outros. É também esta plataforma que

faz a ligação entre o utilizador e os serviços disponibilizados pelo PUC, permitindo que o primeiro usufrua das funcionalidades deste último.

1.5 Estrutura do documento

Este documento encontra-se dividido em 6 capítulos com diferentes propósitos.

O primeiro capítulo introduz de uma forma simples o problema a resolver, como foi abordado, e quais as principais contribuições efectuadas para a sua resolução.

No segundo capítulo detalham-se as tecnologias e conceitos que influenciaram o desenvolvimento da tese de mestrado, de forma dar um melhor enquadramento da solução final e permitindo que o leitor entenda as razões para as escolhas e decisões descritas nos capítulos seguintes.

O terceiro capítulo apresenta a arquitectura da solução proposta, como e por que módulos são compostas as duas plataformas desenvolvidas e como se ligam a serviços externos. O seu objectivo é a disponibilização de uma visão geral das plataformas, sem entrar nos detalhes de implementação.

No quarto capítulo descreve-se com maior detalhe os pormenores de implementação. Após uma pequena introdução das tecnologias que suportaram directamente o desenvolvimento das duas plataformas, traça-se o perfil dos componentes, como se enquadram no sistema, como comunicam com os outros, e quais as funcionalidades que disponibilizam. No final do capítulo descrevem-se pormenorizadamente algumas das principais interacções entre as duas plataformas.

O quinto capítulo apresenta os resultados directos do desenvolvimento, desde a interface de utilização até às medições de tempos das operações efectuadas sobre as plataformas.

No sexto e último capítulo revêem-se os objectivos propostos no primeiro capítulo e faz-se um balanço de quais os objectivos cumpridos, quais os que não foram, e quais as principais razões para tal. São também feitas algumas considerações acerca do rumo que as plataformas desenvolvidas podem tomar após a conclusão da tese de mestrado.

2 Trabalho relacionado

No decorrer deste último ano, primeiro na preparação e depois na escrita da dissertação, foram estudados vários assuntos e analisadas diferentes soluções tecnológicas, de modo a se poder ter uma melhor percepção global do enquadramento desta dissertação de mestrado. Neste capítulo é feita uma introdução a algumas das tecnologias estudadas e analisam-se as suas principais características, com o objectivo de identificar os aspectos que influenciaram o desenvolvimento desta tese.

O primeiro ponto deste capítulo introduz-nos no domínio dos protocolos de transmissão de dados em rede, analisando-se alguns dos mais vocacionados para a troca de vídeo e áudio. É feita uma pequena descrição destes protocolos, detalhando os seus pontos diferenciadores e em que campos levam vantagem em relação aos outros.

De seguida procede-se à apresentação da tecnologia IPTV, explicitando o seu intuito geral e de que forma esta se apresenta em casa dos utilizadores os serviços em si baseados. Descrevem-se alguns dos principais dispositivos disponíveis numa rede doméstica de suporte a serviços IPTV, incluindo as câmaras de rede e as *internet-enabled TVs*. Evidencia-se que são ambos membros das redes domésticas mencionadas no ponto anterior, e é feita uma comparação entre estas novas gerações de dispositivos e os seus respectivos predecessores (as câmaras analógicas e os habituais televisores), analisando as diferenças não só em termos de funcionalidades como de adequação aos objectivos deste trabalho. Neste capítulo referem-se ainda algumas das mais importantes especificações que orientam a criação deste tipo de serviços, oferecendo algum destaque à implementação em que se baseou o produto final desta tese.

Pretende-se assim com este segundo capítulo que o leitor esteja ciente dos conceitos e das tecnologias envolvidas na implementação deste tipo de soluções, que compreenda onde se enquadra a solução apresentada e as razões para algumas das escolhas e decisões tomadas.

2.1 Protocolos de transmissão de media

A troca de áudio e vídeo na *internet* baseava-se até há pouco tempo no esquema *download-and-play*, em que um *media player* era obrigado a transferir a totalidade de um ficheiro de *media* para o poder tocar. À medida que os ficheiros disponíveis foram aumentando de dimensão e qualidade, sentiu-se a necessidade de implementar esquemas de transmissão que permitissem aos utilizadores visualizar o conteúdo desses ficheiros enquanto decorre o *download*, de modo a evitar longos tempos de espera.

2.1.1 HTTP progressive download

O HTTP progressive download é um método de transmissão de dados equivalente ao normal *download* de um ficheiro, mas onde um cliente pode iniciar a visualização dos conteúdos após a transmissão de uma pequena percentagem dos dados, sem ter de aguardar pela totalidade do ficheiro.

O protocolo utilizado para a transmissão é, como o próprio nome indica, o HTTP, pelo que qualquer servidor *web* (como por exemplo o Apache ou o IIS) pode disponibilizar estes ficheiros sem que seja necessária toda uma infraestrutura paralela – a entrega destes ficheiros de vídeo é, do ponto de vista do servidor, igual ao de uma imagem, um documento em PDF, ou qualquer outro tipo. À medida que o ficheiro vai sendo transferido, os conteúdos são armazenados num *buffer* para futura visualização no *player*, podendo o utilizador fazer *seek* para qualquer ponto do ficheiro que já tenha sido carregado. Utilizando este método é impossível fazer *seek* do vídeo para uma posição que equivale a dados que não tenham sido ainda transferidos[1]. Consequentemente, não é também possível saber com exactidão quais as

1 Existem tecnologias derivadas do HTTP *progressive download* como o *Seek streaming* ou *Pseudo streaming* que se baseiam na secção *Range-Request* dos cabeçalhos dos pacotes HTTP e que permitem ao utilizador tocar uma posição temporal no vídeo que não tenha sido transferida ainda (obrigando o cliente a começar uma nova transferência a partir dessa mesma posição). Estes métodos são habitualmente utilizados apenas para vídeos em formato FLV e requerem servidores e *players* específicos para aplicações Flash.

porções do vídeo a que o cliente efectivamente assistiu: se o utilizador desistir de visualizar os conteúdos mas não parar o *download*, todo o ficheiro será transferido.

Os conteúdos transferidos através do *progressive download* são efectivamente guardados em disco. Para o utilizador, isto significa que é simples voltar a tocar conteúdos que já foram tocados, não havendo necessidade de voltar a transferir os mesmos dados. Para quem distribui, significa que é extremamente simples fazer uma cópia dos conteúdos para distribuição ilegal.

2.1.2 Streaming

O *streaming*, ao contrário do *progressive download,* estabelece uma sessão com canais para a transferência dos dados e para a troca de mensagens de controlo entre o servidor e o cliente. Estas mensagens fazem a gestão da velocidade a que os dados são enviados, a qualidade (*bitrate*) do vídeo a ser transferido, ou a posição do vídeo a que o utilizador quer aceder.

A transmissão por *streaming* requer servidores e protocolos apropriados. Existem no mercado vários *streaming servers*, desde soluções *open source* a implementações proprietárias que custam milhares de euros. De entre os vários disponíveis destacam-se o Wowza Media Server (*open source*), Windows Media Server (Microsoft), Flash Media Server (Adobe), Darwin Media Server (Apple, *open source*). Como curiosidade, e adiantando já algum conteúdo reservado para os próximos capítulos, introduz-se a solução escolhida para utilização no desenvolvimento da tese: Wowza Media Server [1], um *streaming server open source* escrito em Java que permite a distribuição de vídeo e áudio de uma fonte única para múltiplos dispositivos (computador pessoal, telemóvel, iPad, ou outro), utilizando vários esquemas de transmissão (todos os descritos neste capítulo – HTTP *progressive download*, *streaming*, *adaptive streaming*), e para várias tecnologias (Adobe Flash, Microsoft Silverlight, Apple Quicktime, entre outros).

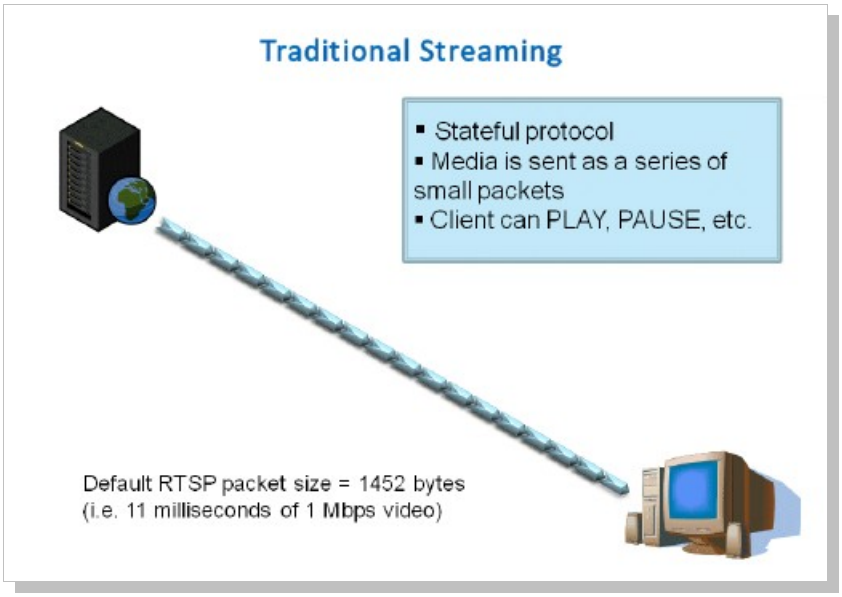

Figura 2: Streaming

Fonte: "IIS Smooth Streaming Technical Overview", Alex Zambelli [2]

Contrariamente ao *progressive download,* o protocolo de rede utilizado por omissão no *streaming* é o UDP, que garante uma maior velocidade na transmissão dos pacotes (à custa da fiabilidade de entrega desses mesmos pacotes. No entanto, sendo o *streaming* vocacionado para aplicações de tempo real, geralmente não há possibilidade de retransmitir dados, pois já não irão a tempo útil para serem aproveitados).

O *streaming* permite iniciar a transmissão em qualquer ponto temporal de um vídeo, ou fazer *seek* enquanto o ficheiro está a ser tocado. Do lado do distribuidor de conteúdos, permite saber com exactidão quais as porções do vídeo mais tocadas, quantos utilizadores estão de facto a assistir ao vídeo num determinado momento, ou por quanto tempo um certo ficheiro foi visualizado.

O facto de apenas serem transferidos os dados que efectivamente são visualizados leva a que a largura de banda necessária seja menor do que no *progressive download.* Adicionalmente, como não existe necessariamente uma sequência ou continuidade na ordem pela qual são transferidos os dados, não é possível guardar o ficheiro em disco (os dados são descartados pelo *media player* após serem visualizados), o que permite ao distribuidor proteger melhor os seus conteúdos, mas obriga à retransmitissão dos dados quando o cliente decide rever porções do vídeo.

2.1.2.1 RTSP

O RTSP (*Real Time Streaming Protocol*) [3] é um protocolo de rede que controla a entrega de dados de vídeo e áudio em ambientes de tempo real, utilizando como fonte *live data feeds* (transmissões em directo) ou conteúdos armazenados em disco. Este protocolo visa gerir o estabelecimento e controlo de sessões e das várias *streams* de dados a eles associados, delegando a entrega dos dados para protocolos desenhados com esse intuito, como é o caso do RTP (*Real-time Transport Protocol*).

> *"In other words, RTSP acts as a "network remote control" for multimedia servers.*
>
> Fonte: "RFC 2326 – Real Time Streaming Protocol (RTSP)", Henning Schulzrinne [3]

O protocolo RTSP é intencionalmente semelhante ao HTTP/1.1 em termos de sintaxe e operacionalidade de forma a que os mecanismos de extensão desenhados para este último possam ser facilmente adaptados ao primeiro. No entanto, há algumas diferenças do RTSP para o HTTP/1.1. A principal diferença é ao nível da gestão de sessões: o RTSP é um protocolo com estado (*stateful*), na medida em que o servidor necessita quase sempre de guardar informação de sessão; enquanto que o protocolo o HTTP não mantém estado (*stateless*). Enquanto que no protocolo RTSP tanto o servidor como o cliente podem executar pedidos, no HTTP só o cliente pode requisitar operações ao servidor. O protocolo RTSP implementa ainda um conjunto de directivas próprias, de entres as quais se destacam:

- `SETUP`: inicializa as variáveis necessárias para a transmissão de um *stream*. Antes de um pedido `PLAY` é necessário fazer um pedido `SETUP` para garantir coerência nas comunicações entre servidor e cliente.
- `PLAY`: estabelece o início (ou o recomeço, caso estivesse pausada) da transmissão dos dados relativos a um ou a vários *streams*. Este pedido pode ser relativo a um *stream* específico ou a todos os *streams* simultaneamente.
- `PAUSE`: interrompe por um tempo limitado um ou todos os *streams*. A transmissão pode ser mais tarde retomada com um pedido `PLAY`.
- `RECORD`: o cliente envia um *stream* ao servidor para que este o guarde.
- `TEARDOWN`: termina uma sessão entre cliente e servidor.

2.1.3 Adaptive streaming

O *adaptive streaming* é um processo que ajusta a qualidade do vídeo transmitido mediante as condições de rede disponíveis de modo a garantir a melhor experiência de visualização possível. Embora se apresente ao utilizador como um normal *streaming*, esta tecnologia baseia-se no HTTP *progressive download*, dependendo deste protocolo para a entrega dos dados. No entanto, ao contrário do HTTP *progressive download*, onde é transferido um único ficheiro com a totalidade do vídeo, no *adaptive streaming* é feita uma pré-codificação do ficheiro que o divide em múltiplos fragmentos. Esta operação dá azo a que entre cliente e servidor haja uma sequência de pequenos *downloads* ao invés de um único mais longo, como mostra a Figura 3.

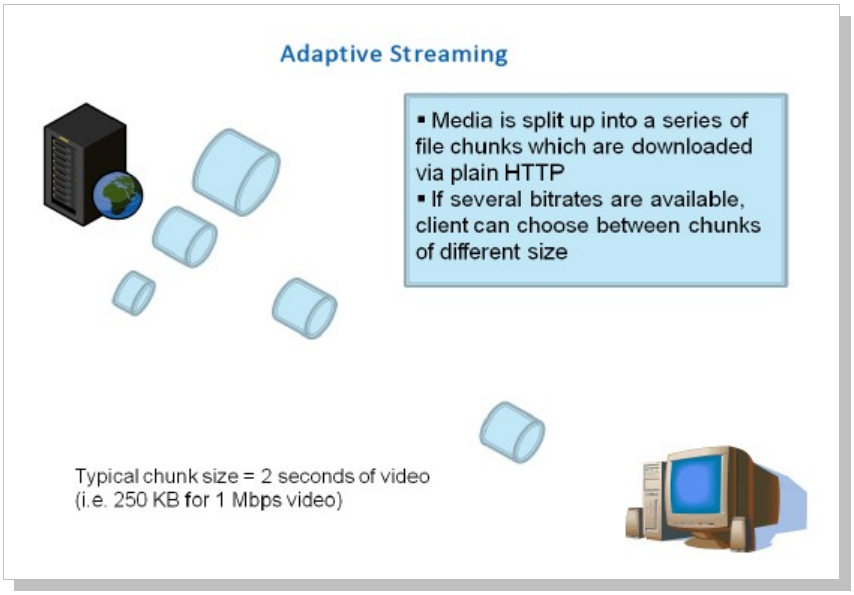

Figura 3: Adaptive streaming

Fonte: "IIS Smooth Streaming Technical Overview", Alex Zambelli [2]

O *adaptive streaming,* por utilizar as infraestruturas tradicionais de HTTP, usufruiu dos mecanismos de *caching* e *proxying* sem necessitar de servidores especializados, sendo desta forma mais barato de implementar do que toda uma rede de *streaming* tradicional. As vantagens para o utilizador são tempos de *startup* e *seeking* mais rápidos, pois estas operações podem ser iniciadas utilizando um *bitrate* mais reduzido e só depois se comutando para uma qualidade mais elevada; e a inexistência de tempos de *buffering* (pausa no vídeo para carregamento de dados) ou *stuttering* (entrega dos dados mais rápida do que a

velocidade de *playback* causa “soluços” no vídeo). Isto é possível desde que a ligação do utilizador suporte os requisitos mínimos de velocidade, por ser esta a determinar a qualidade do vídeo a ser transferido.

2.1.3.1 Microsoft Smooth Streaming

O Smooth Streaming [2][4] é a implementação da Microsoft do esquema *adaptive streaming*. Utilizando pela primeira vez em larga escala com a transmissão *online* dos Jogos Olímpicos de 2008, esta tecnologia entrega pequenos fragmentos de vídeo (habitualmente 2 segundos de vídeo) ao cliente, que verifica se estes chegaram nas condições previstas. Se assim não acontecer, o sistema diminuirá a qualidade de vídeo dos próximos fragmentos até que haja condições para a voltar a aumentar.

Formato dos ficheiros

Durante o evento de estreia desta tecnologia, a principal queixa dos distribuidores de conteúdos foi a dificuldade de gestão dos milhares de pequenos ficheiros criados pela codificação das várias horas de vídeo a transmitir (um ficheiro por cada 2 segundos, multiplicado pelo número de *bitrates* disponibilizados). A equipa da Microsoft mudou então de estratégia e passou a utilizar um único ficheiro em disco por cada *bitrate*, cada um deles contendo os vários fragmentos de vídeo – o formato escolhido foi o MPEG-4 Part 14 (ISO/IEC 14496-12) [5].

A unidade básica de um ficheiro MP4, chamada '*box'*, é constituída por dados e metadados. Na maioria dos cenários de distribuição de *media* é considerado vantajoso colocar os metadados no início do ficheiro, visto que possibilita obter informação sobre os dados que se vão tocar antes mesmo de os começar a tocar. No entanto, nos cenários de distribuição de eventos *live* a maioria dos metadados não são ainda conhecidos (os eventos não aconteceram ainda), pelo que se escolheu escrever os fragmentos como sequências de pares metadados/dados. Este esquema permite também menores tempos de *startup*, visto que há menos informação para processar no início do ficheiro.

Como se verifica na Figura 4, o ficheiro inicia-se com um '*moov'*, que contém alguma informação genérica acerca do ficheiro, aparecendo a partir daí uma sequência de vários

fragmentos, cada um deles constituído por metadados ('*moof*') e dados ('*mdat*'). No final do ficheiro escreve-se um '*mfra*', que contém informação de indexação que permite localizar rapidamente um determinado fragmento.

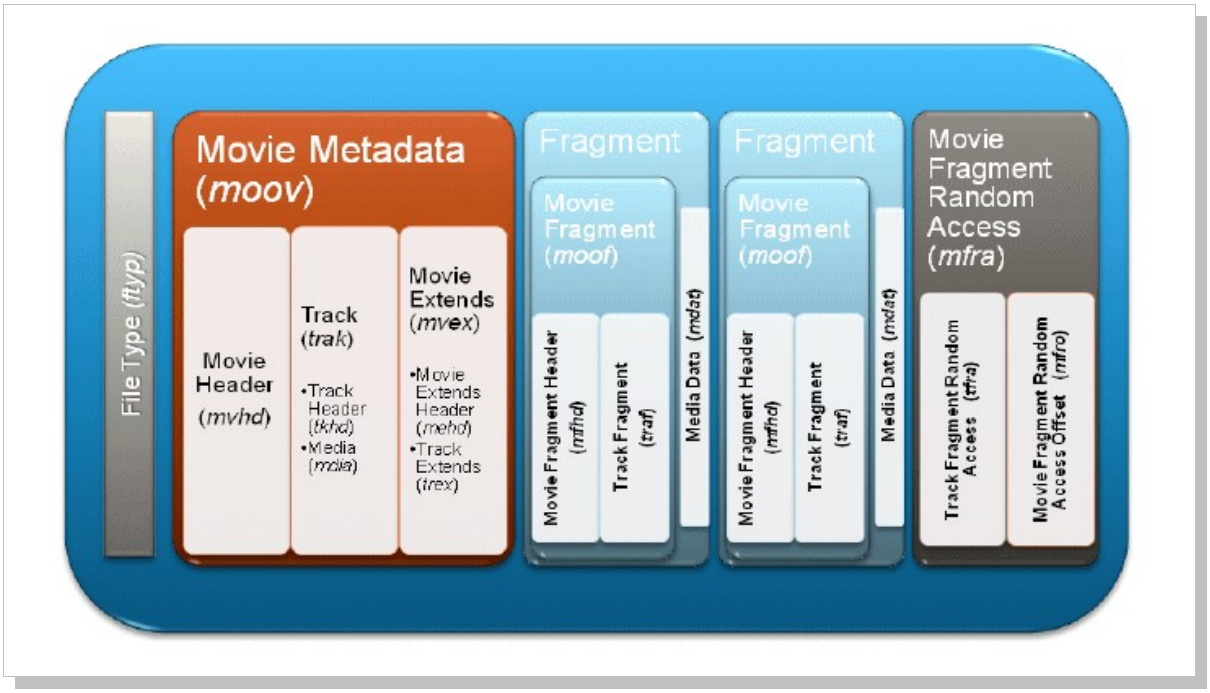

Figura 4: Formato de ficheiro Smooth Streaming

Fonte: "IIS Smooth Streaming Technical Overview", Alex Zambelli [2]

Ficheiros criados

O formato Smooth Streaming utiliza não só uma organização própria para as várias caixas que compõem o ficheiro mas também algumas caixas que não estão delineadas na especificação. De forma a distinguir estes ficheiros dos normais MP4 utilizam-se as extensões **.ismv* (vídeo e áudio) e **.isma* (apenas áudio).

Além destes dois tipos de ficheiros criados em disco, existem também dois *manifest files*: o *client manifest* e o *server manifest*. Os dois ficheiros XML, ambos gerados durante a codificação do vídeo original para o formato Smooth Streaming, têm funções diferentes. O *server manifest* (**.ism*), utilizado apenas pelo servidor, mapeia a relação entre os *bitrates* e faixas de vídeo com os vários ficheiros em disco. O *client manifest* (**.ismc*) é, como o próprio nome indica, enviado ao cliente e contém informações acerca dos *codecs* e *bitrates*

utilizados na compressão, para que este possa inicializar o *decoder* de vídeo correctamente, e também uma lista dos fragmentos de vídeo disponíveis no servidor (juntamente com os seus *start time* e duração).

Protocolo

No início da transmissão, o cliente pede ao servidor o envio do *client manifest* com a descrição do ficheiro e de seguida inicia uma sequência de pedidos para o próximo fragmento que deseja receber através de um URL RESTful[2] com o formato:

```
http://example.com/video.ism/QualityLevels(400000)/Fragments(video=610275114)
```

O servidor pesquisa então no *server manifest* (`video.ism`) pelo *bitrate* pedido pelo cliente (`40000`) de modo a determinar qual o correspondente ficheiro **.ismv* em disco. De seguida, analisando o índice localizado na caixa *mfra* do ficheiro, descobre qual o fragmento que corresponde ao *offset* pedido pelo cliente (`610275114`). Este fragmento (*moof* + *mdat*) é enviado então ao cliente pela rede como um ficheiro *standalone*, e visualizado no cliente.

2.2 IPTV

IPTV (Internet Protocol Television) é um sistema de distribuição de serviços de televisão digital sobre uma rede IP. O IPTV permite a disponibilização de serviços como *live* TV, EPG[3], *content-on-demand*[4] e *time-shifted programming*[5] por parte dos distribuidores de conteúdos, apoiada na infraestrutura de rede da Internet em vez dos tradicionais métodos de entrega do sinal (como por exemplo o rádio, cabo ou satélite), o que possibilita uma maior interactividade entre o utilizador e o serviço.

2 URL que obedece ao esquema REST (*Representational State Transfer*) [6]
3 Electronic Program Guide: guia digital que disponibiliza informação sobre horários de conteúdos televisivos.
4 Disponibilização de conteúdos (filmes, concertos, etc) escolhidos pelo utilizador para visualização imediata ou num futuro próximo, permitindo funcionalidades como *pause/resume*, *rewind* e *fastforward.*
5 Gravação de um conteúdo (habitualmente *live TV*) para um dispositivo de armazenamento para poder ser visualizado a uma hora mais conveniente para o utilizador.

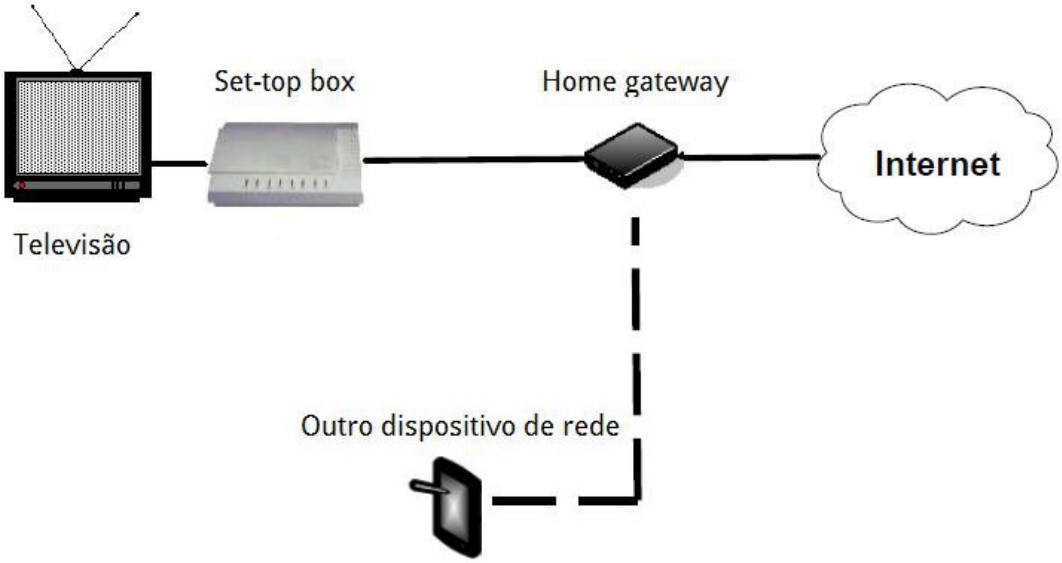

Figura 5: Esquema habitual de IPTV numa rede doméstica

Os serviços de IPTV são entregues ao utilizador por meio de *set-top boxes*, dispositivos que providenciam a necessária ligação à rede IP para receber e descodificar o vídeo para a visualização na televisão. A *set-top box* é também responsável pela gestão dos conteúdos gravados e pode ainda guardar informação sobre o perfil de utilizador.

2.2.1 Componentes

Existem hoje em dia variados dispositivos com capacidades de ligação em rede em casa do utilizador. Não podendo detalhar o funcionamento de todos esses componentes, nesta secção vão ser explicadas as principais características dos dispositivos mais intrinsecamente ligados com o desenvolvimento desta plataforma.

2.2.1.1 Set-top box

Uma *set-top box* é um dispositivo que recebe um sinal digital ou analógico proveniente de um fornecedor de serviço de televisão e converte-o para conteúdo possível de ser disponibilizado num televisor. Pouco populares até há uns anos, o número de *set-top boxes* disponíveis em casa dos utilizadores tem aumentado rapidamente com o crescimento das ofertas de IPTV. Neste tipo de serviço, estas caixas (apresentadas na Figura 6) actuam como

computadores ligados a uma rede IP e providenciam ao utilizador algum nível de interactividade com o fornecedor de serviço.

Figura 6: Set-top box MEO

Fonte: MEO - O comando é meu [7]

As *set-top boxes* extendem muito as funcionalidades da televisão. Equipadas com discos rígidos, estes dispositivos permitem controlar a emissão de televisão, colocando-a em pausa ou rebobinando para o início do programa a decorrer, havendo também a possibilidade de gravar o programa para o rever mais tarde. Suportam ainda uma das funcionalidades mais apreciadas nas ofertas de IPTV: alugar filmes directamente para a televisão do utilizador, sem necessitar de sair de casa.

2.2.1.2 Televisor

Encarada no passado como meros dispositivos para disponibilização de imagens e sons, a televisão é hoje em dia vista como um dos mais importantes elementos de uma solução de entretenimento e multimédia em casa. As mais recentes gerações de aparelhos têm trazido mais e melhores funcionalidades como interfaces USB para discos rígidos externos, memória física para armazenamento de filmes e fotografias, interfaces de rede para ligação à internet, entre outros. A facilidade de integrar com o normal funcionamento da televisão aplicações e conteúdos directamente da internet (vídeos do Youtube, Amazon ou Netflix, fotografias do Picasa, *feeds* de jornais, informação em tempo real sobre a bolsa ou as condições meteorológicas) traz novas possibilidades aos produtores de serviços e aos distribuidores de conteúdos multimédia.

Existem já no mercado algumas soluções de *internet enabled television*, entre as quais as da Panasonic (Viera Cast) [8], Samsung (Internet@TV) [9], LG (NetCast) [10], VIZIO

(Connected TV) [11] ou Sony [12]. Sendo na sua maioria soluções proprietárias, cada uma contempla funcionalidades e conteúdos exclusivos dependentes dos acordos de fornecimento estabelecidos com os vários distribuidores presentes na internet como a Google, a Amazon ou a Netflix.

A Yahoo! tem em curso colaborações com vários fabricantes, entre os quais a Samsung, LG, Sony, VIZIO, e disponibiliza já, para dispositivos destas marcas, *widgets* [13] (Figura 7) que permitem ao utilizadores receber conteúdos de fornecedores como a Amazon, Blockbuster, CBS, Showtime, e USA Today directamente na televisão. Outros *widgets* fornecem funcionalidades como receber *updates* do Twitter e Facebook, seguir acções da bolsa, saber o estado do tempo, ou ver fotografias do Flickr.

Figura 7: Exemplo de widgets do Yahoo! Connected TV

Fonte: Yahoo! Connected TV [13]

Por outro lado, a Skype, empresa produtora de um das mais populares soluções de videoconferência em todo o mundo, iniciou com a Panasonic e a LG uma colaboração que visa integrar nos seus sistemas de software integrados na TV (Viera Cast e NetCast, respectivamente) uma versão da aplicação com o mesmo nome [14] (Figura 8).

Figura 8: Skype na televisão

Fonte: Skype on your TV [14]

Como resultado desta iniciativa, os utilizadores terão à disposição já em 2010 televisores totalmente preparados para fazer chamadas telefónicas de áudio e vídeo utilizando a infraestrutura habitual do Skype, sem que seja necessária a instalação e configuração de dispositivos ou software extra. Estas televisões virão já integradas com câmaras especiais preparadas para a captação de voz a partir de um sofá e com suporte a resoluções de alta definição.

2.2.1.3 Câmara de rede

Câmaras de rede (ou câmaras IP) são dispositivos de captura de vídeo normalmente utilizados em pequenos circuitos fechados e que utilizam a rede IP para a transmissão de dados e sinais de controlo. Nos últimos anos tem-se assistido a um aumento do interesse nas câmaras de rede em detrimento das tradicionais câmaras analógicas, devido a vantagens como:

- Redução de custos: deixa de ser necessária a instalação de uma infraestrutura própria, podendo ser aproveitada uma simples rede IP.

- Melhores capacidades e mais funcionalidades: imagens de alta resolução, mais e melhores formatos de áudio e vídeo, maior capacidade de armazenamento e menor espaço físico utilizado nesse propósito (devido aos melhores algoritmos de compressão e às menores dimensões de um disco rígido de computador), possibilidade de controlar remotamente o sistema a partir de casa.

- Maior flexibilidade: a utilização de câmaras sem fios facilita a colocação de câmaras no local pretendido, não estando limitados por cabos de dados ou energia. A adição, substituição ou remoção de câmaras torna-se também bastante mais simples.

Ao contrário das tradicionais câmaras de vídeo, onde são suportados formatos de vídeo já estabelecidos no mercado (NTSC ou PAL), a flexibilidade das redes IP trouxe alguma confusão no que toca a formatos de vídeo, *codecs* e protocolos suportados. Existem dois principais grupos a trabalhar em especificações para normalizar a produção e utilização deste tipo de dispositivos: o Open Network Video Interface Forum [15] (patrocinada por membros como Axis, Bosch e Sony), e o Physical Security Interoperability Alliance [16], (entre os seus membros encontram-se a GE Security (General Electrics) e a Cisco). Ambos os grupos têm vindo a ganhar notoriedade com a adição de novos membros, e têm já publicada uma versão de produção das suas especificações, pelo que nesta altura não é simples indicar qual dos dois se encontra numa posição vantajosa para se tornar no próximo *industry standard*.

2.2.2 Sistemas de distribuição

Embora baseando-se em conceitos e tecnologias já amplamente testadas, a gestão e disponibilização dos serviços IPTV até à casa do utilizador é um processo algo complexo. A figura seguinte procura dar uma ideia de onde se encontram e como se ligam os vários componentes que integram uma habitual rede de distribuição IPTV.

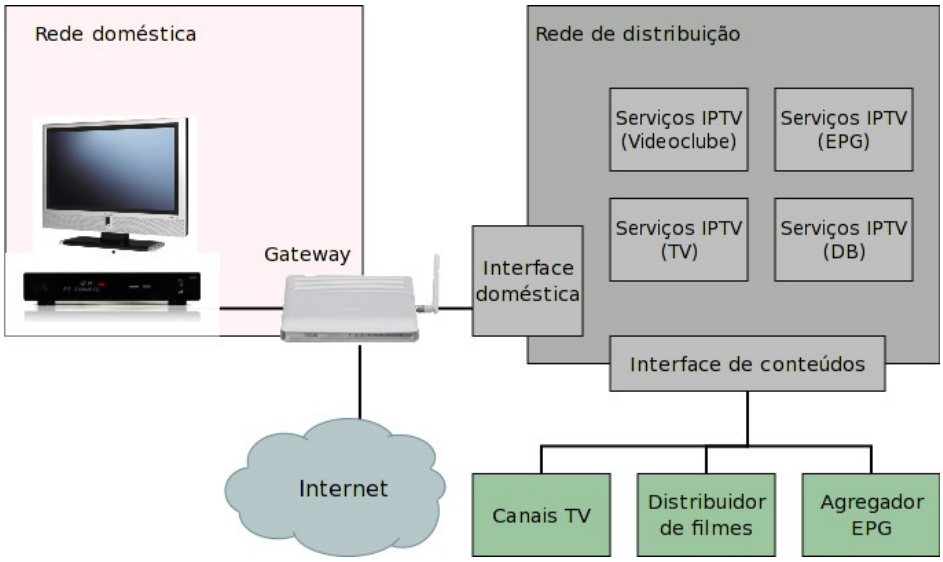

Figura 9: Visão geral da distribuição de serviços IPTV

Existe neste momento um vasto conjunto de soluções de diferentes *players* no mercado que possibilitam uma simplificação dos processos necessários para a disponibilização de serviços IPTV ao utilizador. Nesta secção pretende-se apresentar algumas soluções estudadas no decorrer da tese de mestrado, de forma a que o leitor possa entender algumas das escolhas efectuadas nas fases mais avançadas do desenvolvimento.

2.2.2.1 Open IPTV Forum

O Open IPTV Forum [17] é uma iniciativa de membros chave da indústria cujo intuito é o lançamento de especificações para soluções IPTV que permitam a interoperabilidade das várias soluções, abstraindo-se das tecnologias de transmissão de dados e do tipo de redes usado.

Fundado em 2007 por empresas como a Ericsson, France Telecom, Nokia Siemens, Panasonic Corporation, Philips, Samsung Electronics e Sony Corporation, hoje conta já com mais de 50 membros pertencentes a todos os sectores da indústria, desde a produção de conteúdos aos terminais de visualização, passando pela distribuição e armazenamento desses mesmos conteúdos. Estes participantes têm como finalidade comum a produção de acordos e especificações totalmente abertas e baseadas em tecnologias já existentes que facilitem e aceleram a produção e instalação de soluções IPTV e ajudem a maximizar os benefícios do

IPTV para os vários intervenientes presentes neste mercado: consumidores, operadores de rede, produtores de conteúdos, distribuidores de serviços, produtores de consumíveis, terminais de visualização e até nas infraestruturas de rede.

Existem neste momento vários grupos de trabalho que se propõem trabalhar em áreas relacionadas com o IPTV, vários deles até já com documentação publicada. No entanto, a maioria destes esforços focam-se em áreas específicas (redes domésticas, protecção de conteúdos, etc.) ou cenários específicos de *deployment* (acessos exclusivos em redes protegidas, por exemplo). A iniciativa Open IPTV Forum não procura sobrepor-se a estes grupos já estabelecidos produzindo documentação que substitua a destes, mas sim aproveitar e combinar o trabalho já desenvolvido nesta área para apresentar uma solução global e completa. Na verdade, o Open IPTV Forum estabeleceu já com muitas destas entidades parcerias e acordos estratégicos para que haja possibilidade de existirem trocas de conhecimento e cruzamentos de informação para a produção de melhores *standards* e especificações.

Para que os vários componentes presentes numa solução completa de IPTV se possam complementar e providenciar uma verdadeira experiência de *plug and play*, faz falta a disponibilização de uma ferramenta que permita que facilmente se desenvolvam aplicações que encaixem nessa experiência. Os *standards* criados pelo Open IPTV Forum têm como objectivo ser essa ferramenta.

Arquitectura

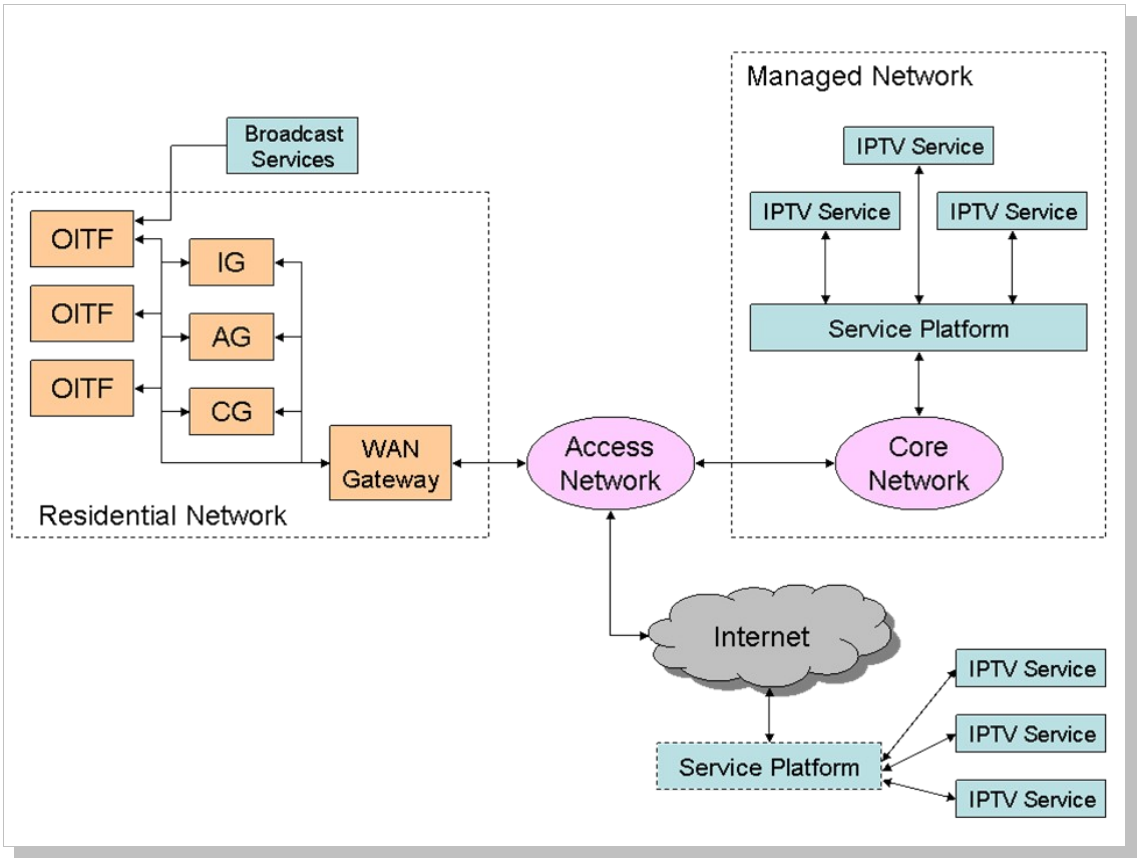

Figura 10: Arquitectura Open IPTV Forum

Fonte: OpenIPTV Release 1 Specification Volume 1: Overview [18]

As especificações do Open IPTV Forum definem dois modelos de acesso aos conteúdos: as *managed networks* e as *unmanaged networks*.

O *managed network model* define as comunicações numa rede controlada ponto a ponto pela empresa de telecomunicações, isto é, o operador gere a produção, agregação e entrega dos conteúdos ao utilizador. Tipicamente, numa rede deste tipo, uma empresa de telecomunicações faz o papel de *service provider*[6], *service platform provider*[7], e *network provider*[8], garantindo assim a qualidade dos serviços entregues ao utilizador. O modelo *unmanaged network* contém os mesmos papéis descritos anteriormente, mas com a diferença de serem protagonizados por entidades diferentes. Verifica-se assim uma clara separação

6 Agrega e prepara os conteúdos, disponibilizando metadados adicionais, DRM, publicidade, etc.
7 Controla o acesso ao serviço de disponibilização de conteúdos.
8 Providencia os recursos necessários para a entrega de conteúdos autorizados ao utilizador.

entre a entidade que disponibiliza os conteúdos e a que fornece o acesso a esses conteúdos. Tipicamente, uma empresa de telecomunicações fornece o acesso à internet, e utilizando essa ligação o utilizador liga-se a um serviço distinto que disponibiliza os conteúdos.

Os *standards* criados pelo Open IPTV Forum dividem-se em dois grupos de trabalho paralelos, chamados *Releases*. Ambas definem a criação de soluções IPTV sobre múltiplas tecnologias de rede, e embora a *Release 1* seja mais focada no acesso tradicional por cabo, faz também uma pequena introdução a alguns outros métodos de acesso à rede. A *Release 2* tem como objectivo cobrir mais detalhadamente o acesso móvel e a disponibilização de serviços IPTV em múltiplos dispositivos como a TV, telemóvel, computador ou PDA.

2.2.2.2 Microsoft Mediaroom

O Microsoft Mediaroom [19] é uma plataforma da Microsoft que permite a criação, gestão e distribuição de serviços IPTV. Esta plataforma é utilizada por vários distribuidores de conteúdos para levar a casa do utilizador um conjunto de funcionalidades como *live TV*, *video-on-demand*, EPG e *scheduled recordings*.

O Microsoft Mediaroom é um update da antiga plataforma Microsoft TV IPTV Edition. Esta mudança de nome reflecte a aposta da empresa na disponibilização de novas experiências a partir da televisão. Assim, apoiado nas funcionalidades clássicas de IPTV, o Mediaroom tem novas capacidades de multimédia, facilitando a visualização de filmes, fotos e músicas guardados em outros dispositivos pessoais. O Mediaroom facilita também ao distribuidor de serviços o desenvolvimento de aplicações que permitam uma experiência interactiva mais rica, como por exemplo portais de compra de vídeos, jogos, e outras funcionalidades.

Figura 11: Microsoft Mediaroom

A solução da Microsoft é encarada como um "ingredient brand" [20], ou um produto de parceria entre a empresa e o distribuidor de conteúdos. Isto significa que, embora o Mediaroom esteja presente desde os servidores de gestão do serviço até ao software em casa do utilizador, será sempre da responsabilidade do service provider a instalação, customização e suporte do produto. Com esta opção, a Microsoft pretende espalhar a plataforma pelo globo e tirar partido do reconhecimento comercial da marca, fazendo com que seja encarada como um factor diferenciador nas ofertas comerciais.

Na casa do utilizador, o Microsoft Mediaroom corre numa *set-top box* dedicada ou na própria consola de jogos da Microsoft, a Xbox 360 (para *service providers* que assim o possibilitem). O Microsoft Mediaroom está representado em Portugal na oferta comercial de IPTV da Portugal Telecom, o Meo [7], e na da Vodafone, o Vodafone Casa [21].

Aplicações

A plataforma Microsoft Mediaroom permite o desenvolvimento de aplicações de utilizador que extendam as suas funcionalidades. Existem três tipos de aplicações disponíveis:

- Microsoft Mediaroom PF application: o Microsoft Mediaroom Presentation Framework é um conjunto de bibliotecas baseadas na tecnologia ASP.NET que permite o desenvolvimento de aplicações de rede completas e escaláveis para a plataforma Microsoft Mediaroom. Estas aplicações podem combinar conteúdo proveniente dos vários clientes e servidores Mediaroom, da *internet*, e de várias fontes privadas (bases de dados, por exemplo). O modelo de desenvolvimento deste tipo de aplicações é semelhante ao modelo ASP.NET, embora tenha um foco diferente: o ASP.NET foca-se na geração de conteúdo HTML, o Microsoft Mediaroom PF foca-se na geração de aplicações que correm directamente na *set-top box*.

- Microsoft Mediaroom Browser application: o Microsoft Mediaroom Browser é um *browser* compatível com o *standard* XHTML 1.0 e DOM que permite aos programadores desenvolverem as suas interfaces de utilização enquanto páginas orientadas para a *web.* Incluído vem também um interpretador de ECMAScript que permite o desenvolvimento de aplicações AJAX ou JSON, e um motor de suporte a CSS para controlo do aspecto da aplicação. O Microsoft Mediaroom Browser é *compliant* com a especificação XHTML 1.0 Strict, e é apenas desenhado para a apresentação de aplicações, não havendo a possibilidade de os utilizadores introduzirem um URL e navegarem por conteúdo fora da rede de serviço.

- RDP application: as RDP *applications* são, como o próprio nome indica, baseadas no protocolo RDP (Remote Desktop Protocol). Os utilizadores acedem a estas aplicações de uma forma semelhante aos outros dois tipos já descritos, mas a aplicação em si executa numa máquina remota, geralmente localizada na rede de serviço. As aplicações RDP podem ser aplicações *web* ou mesmo normais aplicações *stand-alone* para sistemas Microsoft Windows. Está já previsto que o modelo de desenvolvimento deste tipo de aplicações não seja suportado nas próximas versões do sistema Microsoft Mediaroom.

As aplicações que executam na plataforma Microsoft Mediaroom devem obedecer a um conjunto de regras que garantam uma agradável experiência de utilização. Neste sentido, a documentação inclui uma extensa listagem de *guidelines* e *best practices* que indicam aos

programadores como desenhar o aspecto da aplicação, como aumentar a usabilidade e facilitar a interacção com o utilizador final, e também como implementar variados serviços para que tenham uma performance aceitável em sistemas deste tipo.

Interacção com aplicações externas

A plataforma Microsoft Mediaroom disponibiliza às aplicações um conjunto de *web services* SOAP para acesso a várias funcionalidades do sistema, entre as quais gestão de utilizadores, manutenção e listagem de canais, carregamento de EPG e conteúdos de media, e outros. As aplicações que pretendam interagir com o sistema devem construir os seus clientes a partir dos WSDLs fornecidos pela Microsoft e ligá-los aos URLs onde correm os vários *web services* disponibilizados.

Com excepção de documentação interna e informação protegida, a plataforma Microsoft Mediaroom não tem muita informação disponível para consulta, pelo que infelizmente não há possibilidade de descrever mais detalhadamente a sua arquitectura e o funcionamento interno dos seus componentes.

3 Arquitectura da solução proposta

A experiência anterior no desenvolvimento de soluções de colaboração trouxe algumas vantagens na altura da análise das várias soluções possíveis para este problema.

Neste capítulo é primeiramente apresentada a Plataforma Unificada de Colaboração, descrevendo as suas principais características: o que é, para que serve, e que vantagens traz para os seus utilizadores. Uma vez que grande parte do trabalho desenvolvido nesta tese consistiu na reestruturação da implementação do PUC, esta introdução à plataforma é vital para que se compreendam os conceitos e o rumo dos desenvolvimentos descritos nos capítulos que descrevem a concepção e implementação da solução final.

Em seguida introduz-se o novo desenho da arquitectura do PUC, evidenciando as principais diferenças em relação à arquitectura anterior. São apresentados os novos módulos internos que compõem a plataforma, quais as suas principais funções e como comunicam uns entre si.

Neste capítulo é também apresentado o desenho que guiou a implementação do PLAY. É feita uma distinção clara entre os componentes de aplicação (os que utilizam as funcionalidades do PUC enquanto aplicação) e os componentes de recurso (os que se integram no PUC para lhe acrescentar funcionalidades), expondo as razões que levaram a tal separação. A comunicação com os vários serviços externos é também analisada, explicando como se apresenta o PLAY a cada um destes serviços e como são aproveitadas as suas funcionalidades no âmbito dos requisitos da plataforma.

3.1 Plataforma Unificada de Colaboração, versão 1.0

O crescimento que se tem assistido nos últimos anos no número e variedade de dispositivos com capacidades de ligação à internet veio trazer alguma segmentação no mercado das aplicações colaborativas. Funcionalidades como videoconferência, sessões de *chat*, partilha de documentos, *slideshows,* edição colaborativa de documentos, entre outras aplicações, estão implementadas em várias soluções existentes hoje em dia. No entanto, verifica-se que muitas destas aplicações têm problemas de interoperabilidade com outros projectos, ou estão limitadas a certos dispositivos.

Sentiu-se então a necessidade de desenvolver uma *framework* de colaboração que:

- permita a agregação de funcionalidades, provenientes de uma ou mais aplicações, desenvolvidas *in-house* ou por entidades externas ;
- publique as suas funcionalidades na rede, para que outras entidades (operadoras de telecomunicações, por exemplo) as possam utilizar como componentes de uma aplicação ;
- funcione sobre diferentes topologias de rede (TCP/UDP, SIP, RTP, GSM, PSTN) ;
- seja suportada em vários dispositivos (*smartphones*, computadores, televisores) ;

Neste sentido, iniciou-se na PT Inovação, com base no trabalho desenvolvido durante a tese de mestrado do colega Pedro Correia [22], o desenvolvimento da Plataforma Unificada de Colaboração (PUC). Esta plataforma responde a três principais requisitos:

- arquitectura genérica e abstracta
- camada de serviços independente do transporte
- colaboração ubíqua

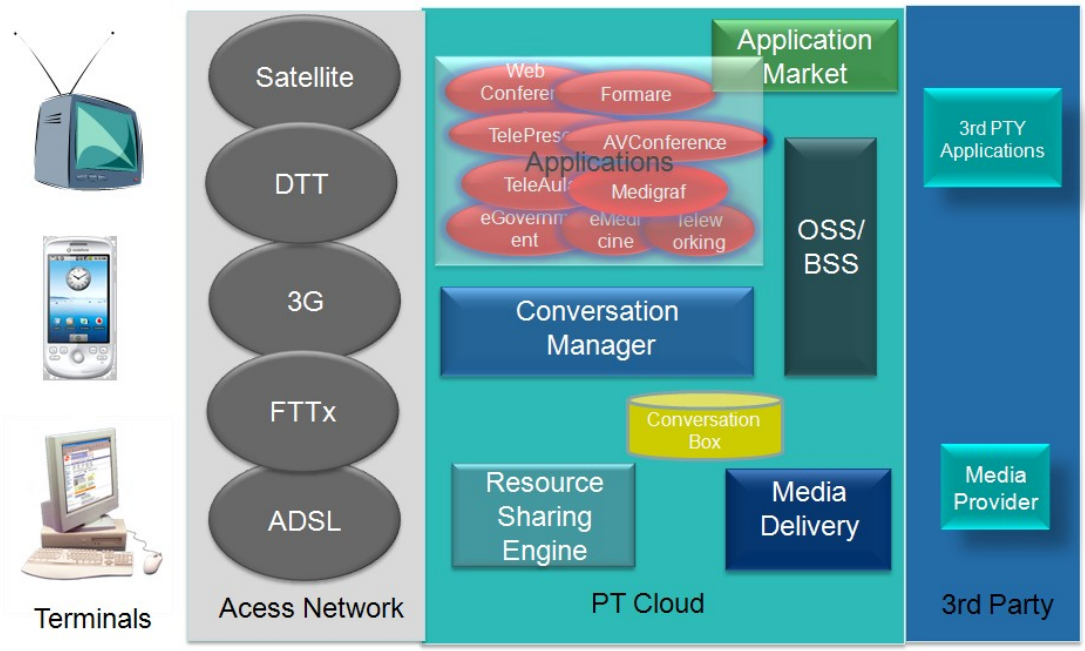

Figura 12: Aspecto geral de uma plataforma colaborativa

Fonte: "A Framework for Collaborative Applications", Pedro Correia [22]

Como se verifica através da Figura 12, os utilizadores podem aceder ao serviço através de diferentes terminais, que por sua vez podem operar sobre várias redes de transporte de dados. A lógica de controlo dos serviços está toda na chamada "*PT Cloud*", um agrupamento lógico de componentes em desenvolvimento no grupo PT cujo objectivo é a implementação de facilitadores (*enablers*) que operem sobre as diferentes redes e, através da disponibilização de interfaces para aplicações externas, permitam que outras entidades possam utilizar e agregar as suas funcionalidades em aplicações de valor acrescentado.

3.1.1 Modelo de dados

A principal unidade de modelação das entidades do PUC é a conversa. Cada conversa é constituída por um grupo que inclui os utilizadores a si associados e um conjunto de sessões. Uma sessão define-se como uma parte de uma conversa ocorrida num determinado intervalo de tempo onde se discute um certo tópico. A plataforma permite que haja vários tipos de sessão (videoconferência, *chat*, *slideshow*, entre outros). A cada sessão está associado um grupo (que identifica os participantes da sessão e é um subconjunto do grupo da

conversa) e vários *ResourceFeatures*, que representam os vários recursos partilhados disponíveis nela.

3.1.2 Arquitectura geral

O desenho do PUC assenta na seguinte estratégia:

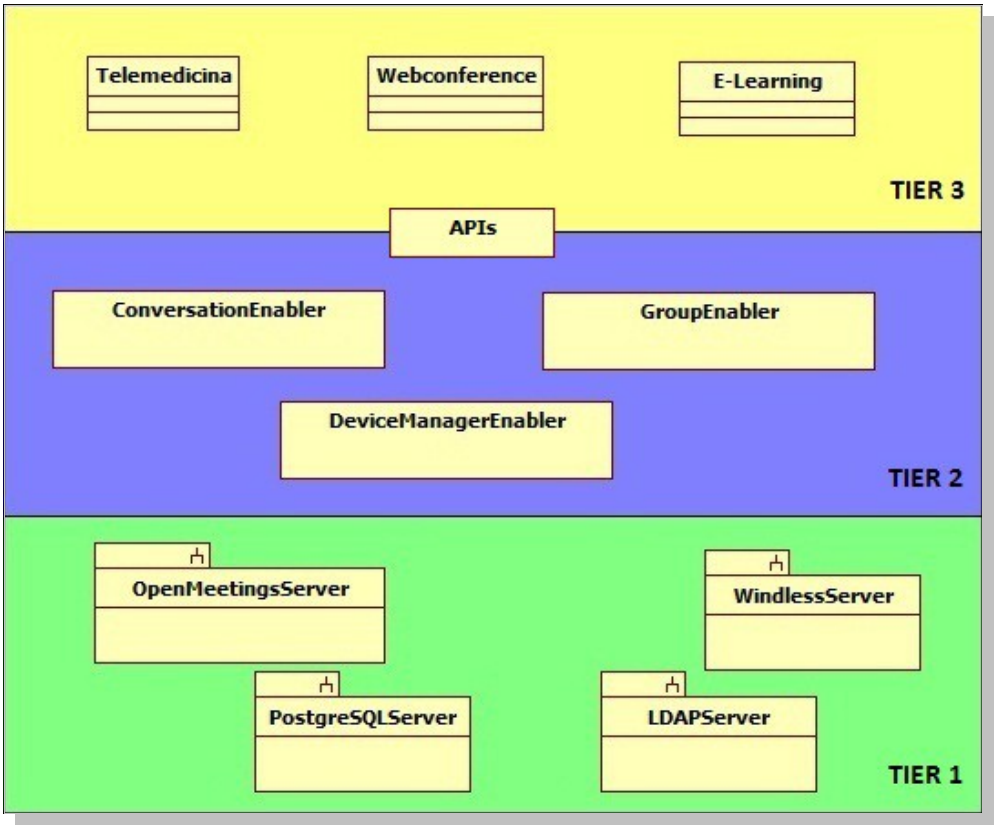

Figura 13: Visão geral da arquitectura do PUC

Fonte: "A Framework for Collaborative Applications", Pedro Correia [22]

Na camada superior encontram-se as várias aplicações desenvolvidas *in house* ou por entidades externas que tenham como objectivo a utilização dos serviços fornecidos pela plataforma através de um conjunto de APIs.

O segundo *tier* é o núcleo do desenvolvimento da plataforma. É aqui que se encontram os vários *service enablers*, módulos independentes que implementam um conjunto de funcionalidades e que permitem controlar os vários recursos disponíveis no sistema.

A camada no inferior da imagem contém os vários servidores externos que irão fornecer ao PUC as funcionalidades de colaboração.

ConversationEnabler

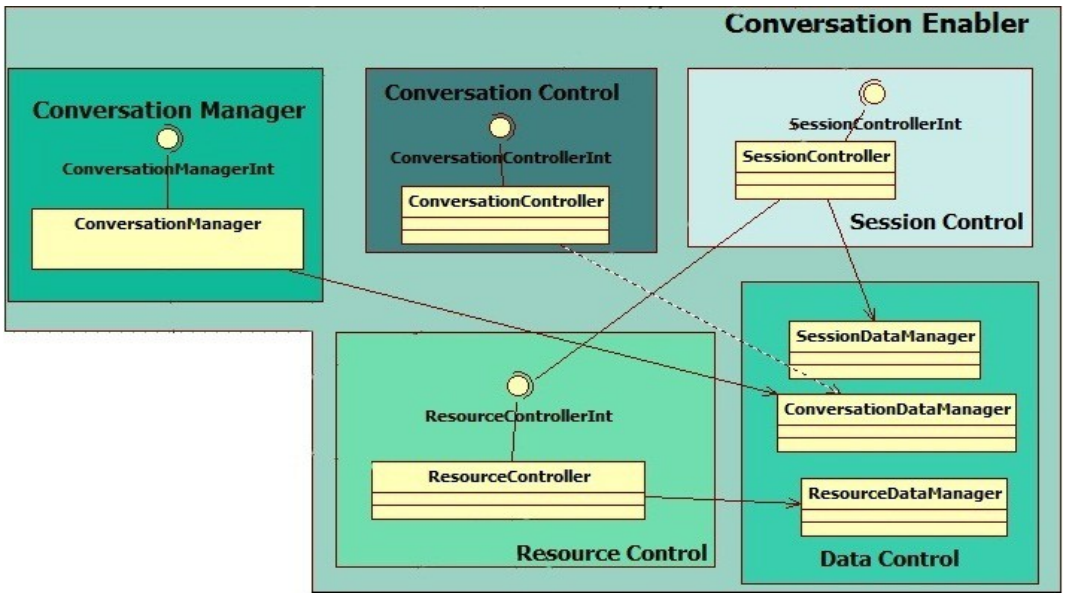

Figura 14: Arquitectura do ConversationEnabler

Fonte: "A Framework for Collaborative Applications", Pedro Correia [22]

Como já foi dado a entender, o `ConversationEnabler` actua para a plataforma como um único módulo, embora seja constituído por vários componentes com diferentes objectivos:

- ConversationManager: gere o conjunto de conversas existentes no sistema, permitindo a criação ou remoção de conversas.
- ConversationController: responsável pelos detalhes específicos de uma determinada conversa. É a este módulo que cabe a tarefa de, por exemplo, gerir a adição ou remoção de participantes e a criação ou remoção de novas sessões.
- SessionController: gere o ciclo de vida das várias sessões pertencentes a uma conversa. Permite a adição ou remoção de participantes, a actualização do estado de uma determinada sessão, e outras funcionalidades.

- ResourceController: controla a comunicação com os vários servidores de recursos que fornecem funcionalidades a uma determinada sessão.
- DataController: facilita a gestão das várias entidades criadas no decorrer de uma conversa e a sua gravação numa base dados externa à plataforma.

GroupEnabler

Este módulo é responsável pela gestão e contextualização dos dados relativos à informação pessoal dos utilizadores da plataforma. Os vários contactos do utilizador (número de telefone, email, *instant messaging*, SIP URI) são tratados de uma forma integrada, permitindo que se adicionem novos contactos à medida que se adicionam mais funcionalidades à plataforma. Cada um destes endereços corresponde a um tipo de recurso e tem associado um estado de presença (Disponível, Ausente, Ocupado) utilizado para construir o contexto agregado do utilizador.

O Group Enabler tem também como o próprio nome indica a responsabilidade de efectuar a gestão dos grupos criados através da plataforma, sejam listas de contactos de utilizador (*buddy lists*) ou listas de participantes de uma determinada sessão ou conversa.

3.2 Plataforma Unificada de Colaboração, versão 2.0

Como acontece em qualquer projecto de desenvolvimento, o primeiro esboço funcional de implementação do PUC revelou alguns problemas, principalmente ao nível dos tempos gastos em algumas operações. Análises aos testes funcionais, de validação e de carga mostraram que essa imperativo proceder a uma reestruturação da organização interna da plataforma de forma a ajustar as interfaces disponibilizadas pelos vários módulos, especificar como se interligam e comunicam entre si, e refinar não só o formato dos dados trocados entre eles, mas também a modelação dos que são passados para a base de dados.

3.2.1 Arquitectura geral

A reestruturação implicou algumas mudanças na arquitectura do PUC, embora se tenha tentando não divergir radicalmente do que tinha sido implementado até então. A Figura 15 apresenta uma visão da nova arquitectura da plataforma.

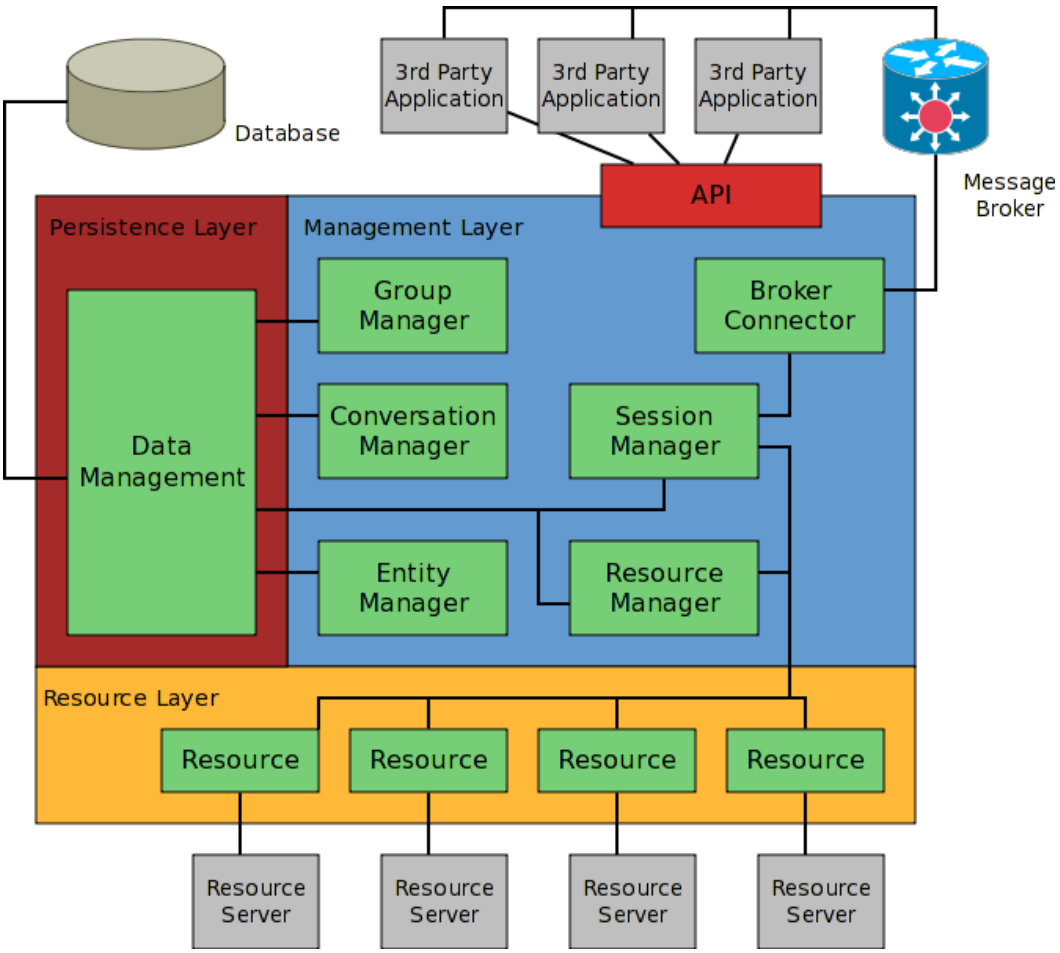

Figura 15: Visão geral da arquitectura do PUC

A noção de *service enablers* enquanto módulos que encapsulam funcionalidades foi bastante alterada, passando a existir ao nível do *tier* 2 (Figura 13) uma divisão bem delineada entre as principais camadas da plataforma: *Management Layer, Persistence Layer* e *Resource Layer*. A gestão de cada funcionalidade inerente à plataforma (conversas, sessões, utilizadores, recursos) deixa de estar encapsulada no antigo *ConversationManager* para ser agora efectuada através do respectivo *manager* (ao nível do *Management Layer*, contém a lógica de negócio) e *service (*ao nível do *Persistence Layer*, contém o código necessário para

a manutenção das várias entidades na base de dados [9]). O *BrokerConnector* faz a ligação a um serviço de entrega de mensagens que permite publicar os vários eventos disparados pela plataforma para que estes sejam consumidos pelas aplicações.

Na *Resource Layer* encontram-se os vários recursos disponíveis, que não são mais que representações na plataforma de servidores externos que adicionam funcionalidades. Cada um destes servidores é gerido por um módulo constituído por dois componentes: um *Controller,* que comunica com a camada superior (*Management Layer*); e um *Connector,* que estabelece a ligação de rede com o servidor externo.

3.3 PLAY - Terminal IPTV

O PLAY, terminal IPTV para a Plataforma Unificada de Colaboração, é uma solução que disponibiliza aos utilizadores da plataforma Microsoft Mediaroom presente no Meo (embora não esteja limitada a apenas esta) uma interface simples e prática que lhes permita aceder aos serviços colaborativos implementados pelo PUC. Utilizando apenas os dispositivos a que está habituado, como por exemplo a *set-top box* ou o controlo remoto, e com uma interface integrada no normal serviço de televisão, o utilizador pode participar em sessões de colaboração com outros utilizadores, ligados à plataforma através de vários tipos de terminais (telemóvel, computador pessoal, telefone fixo ou terminal IPTV).

Como já foi introduzido em 2.2.2.2, a maioria das aplicações que são integradas com as *set-top boxes* compatíveis com o Microsoft Mediaroom não são mais que *links* para canais de TV configurados para apresentar conteúdo *web* dinâmico. Deste modo, a interface de utilizador desta plataforma será constituída por um conjunto de páginas *web* que obedecem aos formatos e normas definidas pela Microsoft, geradas de forma dinâmica e disponibilizadas ao utilizador através da componente de apresentação do PLAY.

Do ponto de vista do PUC, o PLAY funciona simultaneamente como aplicação externa (*3rd party app*) e como servidor de recursos. Funciona como aplicação externa pois utiliza as suas APIs para obter informação acerca dos utilizadores registados, conversas e sessões a

9 Para simplificar a leitura da imagem, na Figura 15 todos os *service e* DAOs (módulos que fazem a representação da plataforma na base de dados – mais detalhado em 4.2) estão apresentados num único componente, *DataManagement*

decorrer, entre outros. Como servidor de recursos, informa o PUC das várias acções que os utilizadores podem tomar, como por exemplo juntar-se a ou deixar uma sessão, iniciar ou parar a partilha de vídeo. É deste modo natural encontrar na arquitectura da plataforma (Figura 16) uma divisão clara entre a lógica de aplicação (módulo *Application*) e a lógica de recurso (módulo *Resource*). Ambos os módulos partilham uma ligação à componente de apresentação (módulo *Presentation*), pois as restrições de funcionalidades ao nível da plataforma Mediaroom obrigam a que todos os conteúdos seja apresentado de uma forma unificada e oriunda de uma mesma fonte. Como se verá mais adiante em 5.1, a página principal de sessão do PLAY irá apresentar elementos que dizem respeito à lógica de aplicação (listas de participantes e recursos, eventos, entre outros) e simultaneamente também permitirá ao utilizador despoletar comandos de recurso (iniciar partilha de vídeo e controlar *slideshows*, por exemplo).

3.3.1 Arquitectura geral

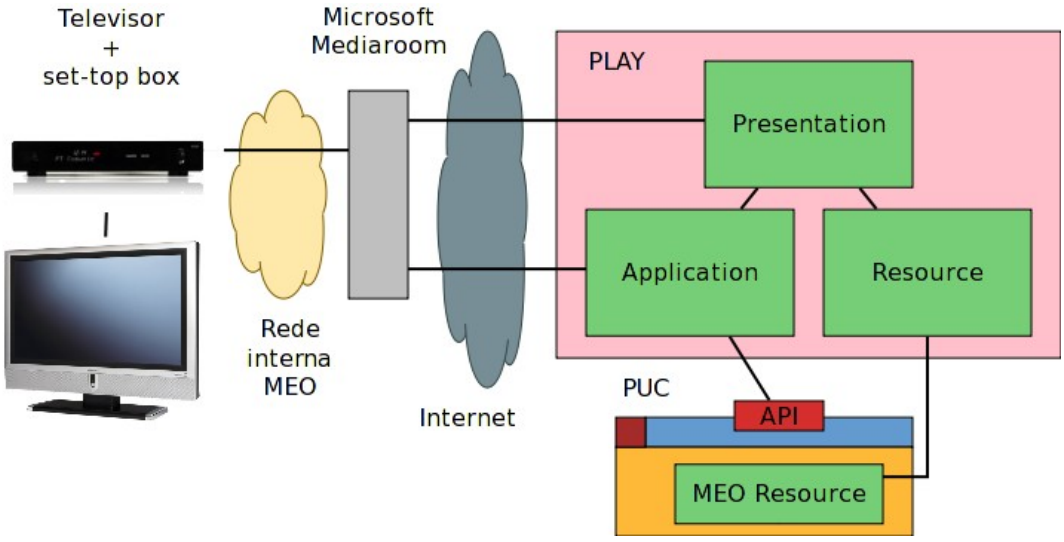

Figura 16: Visão geral da arquitectura do PLAY

A aplicação trabalha sobre a plataforma da PT Inovação em dois níveis distintos: como aplicação e como recurso.

Presentation

O *Presentation* é o ponto de entrada do utilizador na aplicação, gerando a interface visível na televisão. É este componente que recebe os pedidos do utilizador e gera os conteúdos apresentados como resposta, controlando e agregando as funcionalidades dos dois outros módulos.

Application

O módulo *Application* tem duas principais ligações a duas diferentes plataformas: PUC e Microsoft Mediaroom.

Sendo a única informação de identificação do utilizador disponibiliza pela *set-top box* o seu próprio identificador, este componente tem como primeira função utilizar a API da plataforma Mediaroom de modo a identificar o utilizador ao qual a caixa pertence. É também da sua responsabilidade a invocação do serviço de notificação de clientes para que o utilizador correcto seja avisado do início de uma sessão em que está registado.

Do lado contrário, a ligação ao PUC faz-se através do conjunto de APIs estabelecidas para as aplicações e serve para receber o mais variado tipo de informações necessárias ao funcionamento da aplicação: dados do utilizador, conversas e sessões iniciadas, ou recursos disponíveis.

Resource

O módulo *Resource* liga-se à camada de recursos do PUC, actuando para este como um terminal. É através deste módulo que a plataforma é informada de qualquer acção referente a um terminal. Deste modo, acções como a entrada ou saída de uma sessão, o início ou fim de partilha de vídeo com outros utilizadores são comunicados à plataforma pelo *Resource*.

3.3.2 Principais funcionalidades

Operando sobre um terminal historicamente mais orientado à visualização de conteúdos recebidos do que à introdução de novos, a aplicação foi desenhada tendo em mente um papel relativamente mais passivo do utilizador.

Neste sentido, é possível visualizar as conversas e sessões em que o utilizador se registou, podendo este a qualquer altura juntar-se à que desejar. Dentro da sala, o utilizador recebe informação sobre os restantes participantes que se encontram ligados e com que terminais o estão a fazer. Com o pressionar de um botão no comando da MEOBox é apresentada uma lista dos recursos disponibilizados (vídeos, documentos, apresentações ou fotografias). Toda esta informação é actualizada em tempo real através dos eventos recebidos do PUC.

O utilizador é informado assim que é dada instrução para começar uma apresentação através do PUC. O apresentador, que opera em qualquer outro terminal compatível com o PUC, procederá à normal mudança de slides, reflectida no terminal de televisão. No fim do *slideshow* o utilizador recebe uma notificação de que terminou a apresentação.

A aplicação possui ainda funcionalidades de partilha de conteúdos, não se cingindo apenas à visualização dos dados recebidos. Tendo à disposição uma câmara de rede, a aplicação permite publicar no PUC o endereço da câmara, dando hipótese de partilhar com outros terminais com mais capacidades o vídeo do utilizador.

4 Implementação

A implementação das plataformas descritas nesta tese é um processo evolutivo com base em conceitos conhecidos e tecnologias sobejamente utilizadas no desenvolvimento de software.

Este capítulo começa por fazer uma pequena descrição das principais tecnologias utilizadas durante a elaboração da tese de mestrado, introduzindo o leitor a alguns dos termos utilizados no resto do capítulo e aos modelos de programação que ditaram muitas das decisões tomadas durante o desenvolvimento destas aplicações.

Após este primeiro passo introdutório, passa-se a apresentar o modelo de implementação das duas plataformas desenvolvidas durante a elaboração da tese de mestrado, o PUC e o PLAY. Tendo como base as arquitecturas de sistema detalhadas no capítulo anterior, parte-se para uma descrição pormenorizada da implementação, apresentando alguns dos principais componentes dos sistemas e como se ligam entre si, e justificando algumas das decisões tomadas relativamente ao desenvolvimento destes módulos.

4.1 Tecnologias

O desenvolvimento das plataformas descritas neste relatório teve suporte num conjunto de variadas tecnologias. A experiência ganha no passado com a implementação de serviços semelhantes revelou-se uma vantagem, tanto ao nível do desenho e da arquitectura da solução final como da implementação do código que os complementa. No seguimento deste raciocínio, e como tem vindo a ser hábito na PT Inovação, deu-se preferência à utilização de tecnologias já implantadas no mercado empresarial e que se revelam *industry standards*.

Esta primeira secção do capítulo de implementação tem então como propósito introduzir o leitor a algumas destas principais técnicas, *frameworks* e soluções utilizadas no desenvolvimento de ambas as plataformas, de forma a que mais facilmente se possam entender algumas das escolhas tomadas e para que se melhor se percebam alguns dos problemas encontrados no decorrer da tese.

4.1.1 Java EE

Java EE, ou *Java Platform, Enterprise Edition [23]*, é uma plataforma que visa reduzir a complexidade de desenvolvimento de aplicações Java orientadas para serviços empresariais, disponibilizando aos programadores um conjunto de APIs para o *deployment* de serviços *multi-tier*, distribuídos e modulares.

A versão 5.0 da plataforma (*Java EE 5*) tem como foco a facilidade de desenvolvimento, melhorando os processos iniciados pela versão anterior (*J2EE 1.4*) e fazendo uso da nova especificação da linguagem Java (*J2SE 5.0* [24], *Java 2 Platform, Standard Edition 5.0*, vulgo *Java 5*) com especial atenção à tecnologia de *anottations* [25] que esta passou a incluir. Neste sentido, a *framework* inclui novas funcionalidades como *EJB 3.0* (detalhados em 4.1.2), a definição de uma API (*JAX-WS 2.0* [26]) e um conjunto de anotações para a criação de *web services* [27] ou o mapeamento de objectos Java para XML (*JAXB 2.0* [28]).

As aplicações desenvolvidas para a plataforma Java EE 5 são suportadas por um *application server* que tem como função a execução eficiente dos vários módulos que as compõem. É ao nível dos *application servers* que estão localizadas as implementações das especificações definidas pela plataforma (descritas no parágrafo anterior), e são eles que providenciam de forma transparente para a aplicação serviços como *clustering*, *fail-over* e *load-balancing*, retirando ao programador a responsabilidade de desenvolver estes mecanismos.

Para o desenvolvimento das plataformas de que trata este relatório foi escolhido o *JBoss Application Server* (JBoss AS), uma implementação *open-source* desenvolvido pela Red Hat. Este é um dos *application servers* mais utilizados a nível mundial, devido à extensa lista de funcionalidades que oferece e à impressionante quantidade de suporte que recebe

quer por parte da comunidade como por parte dos próprios *developers*. A versão 5.1.0-GA é totalmente compatível com a plataforma Java EE 5, introduzindo já algumas das funcionalidades da próxima versão, Java EE 6 (que, como curiosidade, será totalmente suportada na versão 6.0 do JBoss AS).

4.1.2 EJB 3.0

EJB 3.0, *Enterprise Java Beans* 3.0 [29] é um modelo que especifica o desenvolvimento de componentes *server-side* que implementam lógica de negócio (*beans*) e como estes devem interagir com o *application server* em que estão a executar. A especificação EJB 3.0 é um melhoramento substancial da anterior especificação (EJB 2.0), que por ser extremamente complexa, afastou vários programadores. O seu principal objectivo tornou-se assim a simplificação de processos, facilitando a implementação de componentes e a sua utilização por parte de outros, dependendo agora para a maior parte dos processos de anotações em vez de complicados ficheiros de configuração.

A especificação determina dois principais tipos de *beans*: *session beans* e *message-driven beans*. Os *message-driven beans* são objectos que respondem a eventos despoletados e não chamadas de clientes – não são utilizados objectos deste tipo nas plataformas descritas neste relatório, pelo que não se dará ênfase à sua descrição. Os *session beans* são objectos que representam um único cliente dentro de um *application server*, e dividem-se em dois tipos:

Stateful session beans

Stateful session beans são objectos com estado, associados a um único cliente que originalmente os instanciou. Estes *beans* garantem não só que os dados guardados pelo cliente são armazenados consistentemente mas também que apenas o cliente que os guardou pode aceder a eles. Deste modo, existe uma (e apenas uma) instância de um *stateful session bean* por cada cliente que os utilize.

A implementação de um *stateful session bean* é feita da seguinte forma:

```
@Local
public interface ExampleLocal {
	public int add(int x) ;
}

@Stateful
public class ExampleBean {
	private int count ;

	public int add(int x) {
		count += x ;
		return count ;
	}
}
```

Stateless session beans

Stateless session beans são objectos que, como o próprio nome indica, não têm estado associado. Estes *beans* funcionam geralmente através de um mecanismo de *polling*: quando a aplicação é iniciada, o *application server* executa uma pré-instanciação de um conjunto limitado de *stateless beans*, e a cada chamada de um cliente o AS determina aleatoriamente qual deve responder. Desta forma, a mesma instância destes *beans* pode ser utilizada por vários clientes, embora nunca concorrentemente (estes beans são sempre *thread-safe*).

A implementação de um *stateless sesion bean* é muito semelhante à de um *stateful:*

```
@Local
public interface ExampleLocal {
	public int add(int x, int y) ;
}

@Stateless
public class ExampleBean {
	public int add(int x, int y) {
		return x + y ;
	}
}
```

A especificação EJB 3.1 introduz ainda um terceiro tipo de *session bean*, os *singleton*. *Singleton session beans* são *beans* que têm um estado global partilhado em todo o *application server*, sendo utilizada sempre a mesma instância para responder a qualquer pedido de qualquer cliente. Este tipo de *bean* revela-se extremamente útil para, por exemplo,

a comunicação com os servidores externos utilizados pelo PUC. Uma vez que a especificação EJB 3.1 não é ainda suportada pelo JBoss AS 5.1 (sê-lo-á na versão 6.0), utiliza-se nas plataformas aqui descritas uma extensão proprietária da JBoss que implementa a mesma finalidade: os *service POJOs* (implementados com a anotação `@Service`). É importante frisar que, embora tenham um nome semelhante, estes *service POJOs* não estão de forma alguma relacionados com os componentes de tipo *service*, descritos no próximo capítulo.

4.1.3 Persistência

JPA (*Java Persistence API)* [30] é uma *framework* que determina o conjunto de APIs para a gestão da persistência e do mapeamento objecto/relacional para ambientes Java EE ou Java SE. Originalmente incluída na especificação EJB 3.0, tornou-se num documento totalmente independente com a chegada da versão EJB 3.1. Na JPA define-se como se constroem objectos com estado persistente (*persistence entities*), como devem convertidos para e armazenados numa base de dados relacional, e a linguagem de pesquisa de entradas nessas mesmas base de dados (JPQL, *Java Persistence Query Language*).

As *persistence entities* não são mais do que classes Java cujo estado é armazenado numa base de dados relacional. Estas entidades derivam historicamente dos *entity beans,* presentes nas primeiras versões da especificação EJB (1.0 e 2.0). Cada *entity* pode definir relações com outras *entities*, expressas por anotações ou num ficheiro descritor XML separado.

As soluções relacionadas com persistência utilizadas pelas plataformas tratadas neste relatório são o *Hibernate* [31], implementação *open source* da especificação JPA incluída no JBoss AS 5.1, e o *PostgreSQL* [32] versão 8.4, uma das bases de dados (também *open source*) mais utilizadas a nível mundial.

4.2 PUC

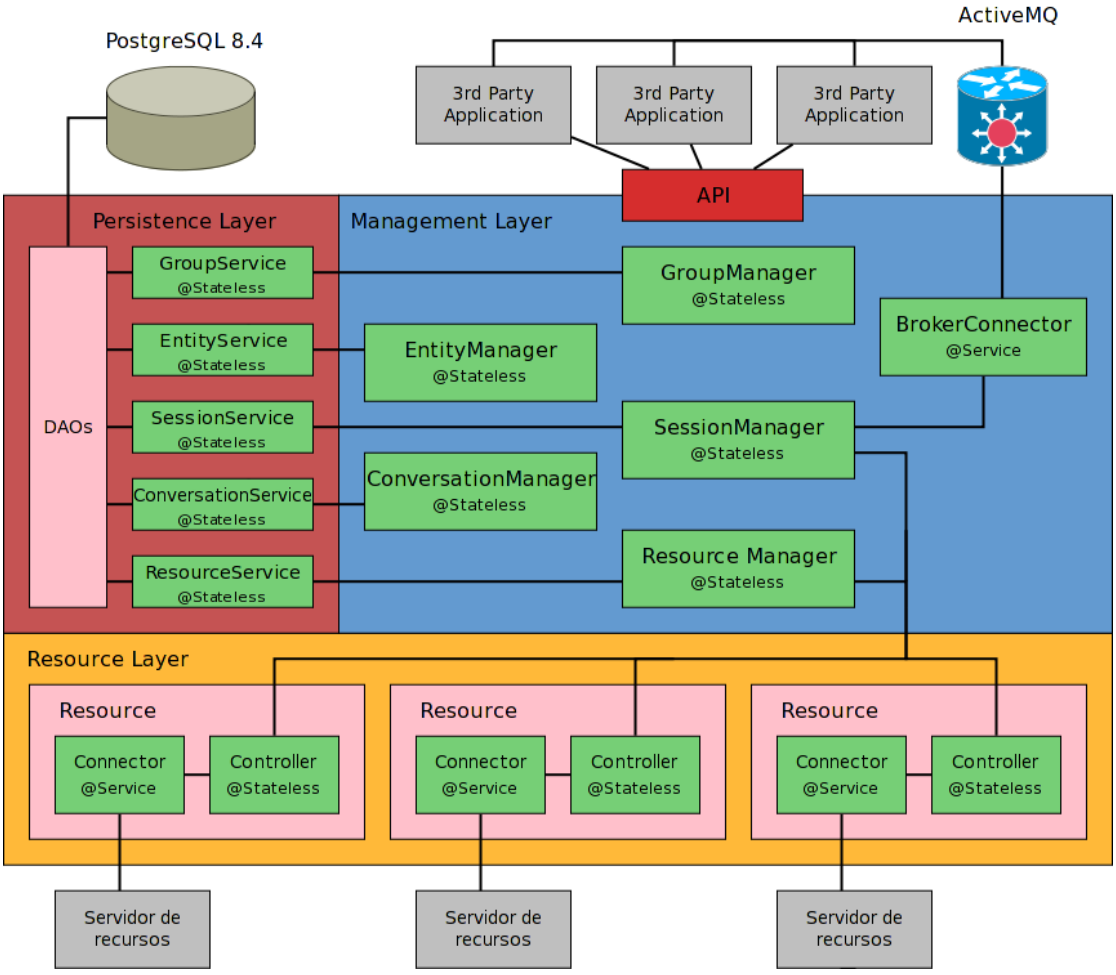

Figura 17: Modelo de implementação do PUC

Como já tinha sido introduzido em 3.2, o desenho em três camadas facilita a abstracção por parte dos componentes presentes no *Management Layer* dos detalhes de persistência dos dados e da lógica específica de cada recurso implementado no *Resource Layer*.

4.2.1 Management Layer

Na *management layer* encontra-se grande parte da lógica de negócio da solução, estando nela contidos dois principais tipos de componente: os *managers*, e o conector JMS.

Os m*anagers,* implementados como *stateless beans,* agrupam funcionalidades relativas a cada uma das principais entidades do PUC. São estes componentes que contêm a *business*

logic da solução, trabalhando sobre as várias entidades e coordenando a sua utilização ao nível da base de dados (através dos vários *services* contidos na *persistence layer*) e ao nível dos recursos (através dos vários *resource controllers* na *resource layer).* São também estas entidades que executam os pedidos provenientes das aplicações. Neste sentido, a API exposta às aplicações externas é constituída por um sub-conjunto das funcionalidades que os *managers* implementam para utilização interna.

A comunicação de novos eventos às aplicações é feita através de um serviço de mensagens assíncronas baseado na especificação JMS (*Java Message Service*) [33], o Apache ActiveMQ [34]. Este serviço, passível de ser totalmente integrado com o JBoss AS, permite a criação dinâmica de tópicos ou filas de mensagens (um por cada sessão a decorrer no PUC) às quais se ligam as aplicações de forma a receber notificações disparadas pelo PUC. A comunicação entre a plataforma e o serviço é efectuada através do `BrokerConnector`. A implementação deste componente, assim como a análise e a manutenção do servidor ActiveMQ foram parte do trabalho desenvolvido durante a elaboração da tese de mestrado do Bernardo Ferreira junto da PT Inovação [35].

4.2.2 Persistence Layer

Na *persistence layer* encontra-se a representação física das várias entidades utilizadas na plataforma (como por exemplo `Conversation`, `Session`, `Participant`) e o código que faz a gestão destas entidades num sistema externo de base de dados. Como também acontece ao nível dos *managers*, cada módulo encapsula as funcionalidades associadas a uma determinada entidade.

Para facilitar a visualização, na Figura 17 estão apenas representados os vários *services* presentes na plataforma. Na verdade, cada um destes *services* controla um agrupamento lógico de DAO (*Data Access Object*), módulo que gere a persistência de uma única entidade na base de dados. Como exemplo, o `SessionService`, controla a execução de outros DAOs relacionados com ele: `SessionDAO` e `ParticipantDAO`. Desta forma garante-se uma separação de funcionalidades, de forma a que os *managers* estejam abstraídos dos detalhes de implementação de cada base de dados e possam conter apenas o código necessário à execução genérica de cada função.

4.2.3 Resource Layer

Ao nível do *resource layer* encontramos os recursos, entidades lógicas que adicionam funcionalidades ao PUC. Cada recurso é constituído por um *controller,* que contém a lógica de aplicação e liga o recurso à camada superior, e um *connector,* que serve de interface entre o recurso e o servidor externo.

Cada *controller* estende uma classe abstracta `ResourceControllerAbstract` e implementa uma interface comum, `ResourceController`, o que lhe permite ser controlado de forma genérica pelos *managers* da camada superior. Embora devam implementar apenas as funções que se adequam ao funcionamento do seu servidor externo, há funções que são obrigatórias a todos os recursos por serem necessárias ao funcionamento da plataforma.

A plataforma tem definidas um conjunto de interfaces que agrupam funcionalidades de cada tipo genérico de servidor: `StreamResourceController`, `MediaResourceController`, `TextResourceController`. Estas interfaces não se referem especificamente a um único servidor de recurso, mas sim a um conjunto de servidores do mesmo tipo (servidores de *media,* de *chat,* em tempo real), sendo através delas que é feita a comunicação com a camada superior do PUC – os *managers* utilizam funcionalidades dos recursos de uma forma genérica ("servidor de *streams*") e não individualmente ("Wowza Media Server"). Quando um dos *managers* chamada uma funcionalidade de recurso, o respectivo *controller* comunica com o *connector* para que este informe o servidor externo das acções a tomar. No sentido inverso (*resource layer* para *management layer*) a sequência é semelhante: cada acção despoletada num servidor de recursos é comunicada ao *connector*, que vez passa a informação ao *controller*, que faz o tratamento dos dados recebidos e chama a função `processEvent()` do `SessionManager`. Esta função executa a lógica necessária para processar a acção na plataforma, incluindo criar um novo evento que será disparado para as aplicações externas através do `BrokerConnector`.

Os *connectors*, como o próprio nome deixa antever, são os módulos que implementam a ligação física ao servidor de recursos. Por serem implementados como *service beans*, garante-se a sua unicidade no sistema, sendo deste modo possível manter um estado e guardar dados específicos a cada ligação. Cada *connector* tem à disposição um vasto leque de protocolos para executar a comunicação com o servidor externo (por exemplo: HTTP, RMI,

SOAP) – a plataforma não especifica a interface com os servidores externos para que cada recurso possa implementar o formato que mais se adeqúe às suas capacidades.

4.3 PLAY - Terminal IPTV

Como já foi introduzido em 3.3, o PLAY apresenta-se ao PUC como aplicação externa e como recurso. De seguida apresentam-se alguns detalhes de implementação da plataforma.

4.3.1 Aplicação

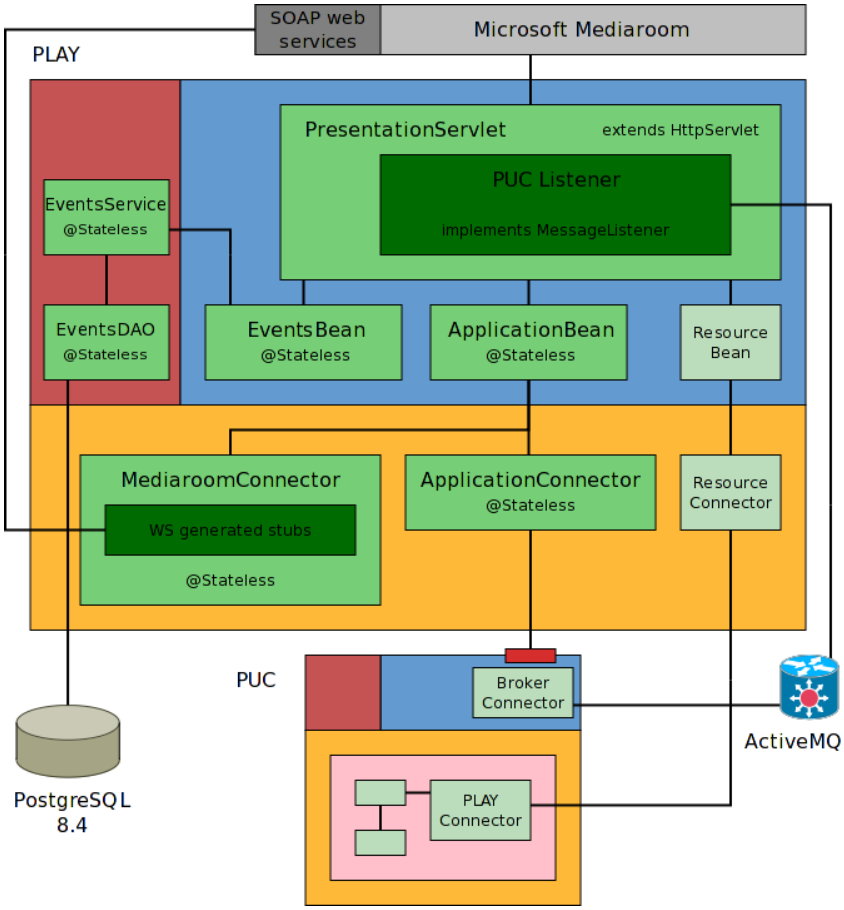

Figura 18: Implementação do PLAY enquanto aplicação

A estrutura da componente de aplicação do PLAY pode, tal como o PUC, ser dividida logicamente em duas camadas: a camada superior, com lógica de negócio; e a camada inferior, com conectores para sistemas externos.

Na camada superior encontra-se a PresentationServlet, que disponibiliza a interface de aplicação para os utilizadores do serviço Mediaroom. A PresentationServlet é responsável por toda a apresentação gráfica da plataforma na *set-top box* do utilizador, através de um conjunto de ficheiros JSP [10] e Javascript. Além de responder directamente aos pedidos dos utilizadores através de normais pedidos HTTP GET ou de acções despoletadas pelos mecanismos de Javascript, este módulo mantém para cada utilizador uma sessão HTTP, que garante não só a coerência do estado de cada participante, mas facilita também a passagem de informação entre o utilizador e a *servlet*. A PresentationServlet tem ainda a seu cargo a manutenção dos vários MessageListeners (um por cada sessão em utilização), objectos que ligam a aplicação ao servidor JMS e permitem desta forma consumir eventos disparados do PUC ("utilizador juntou-se à sessão", "novo recurso disponível", por exemplo) para os apresentar de forma consistente na interface de utilizador. Estes eventos recebidos do PUC são transformados em objectos Event e armazenados numa base de dados própria.

Embora também seja necessária a existência de alguma lógica do seu lado, a PresentationServlet utiliza as funcionalidades dos três *beans* que dividem entre si a maior parte da lógica da solução:

- ApplicationBean: agrega e compõe a informação necessária ao funcionamento do PLAY enquanto aplicação recolhida através dos vários conectores e a entrega à *servlet*
- EventsBean: gere as transacções entre o PLAY e a sua base de dados, armazenando e pesquisando com recurso a vários critérios os eventos recebidos através do PUCEventListener
- ResourceBean: troca informação com o PUC. O seu funcionamento é explicado mais detalhadamente em 4.3.2

A camada inferior contém os vários componentes que ligam o PLAY a serviços externos. Neste momento estão implementados três conectores:

10 JSP, *JavaServer Pages* (JSR-245 [36]), é uma tecnologia que facilita a apresentação de conteúdos dinâmicos orientados para a *web*, semelhante ao ASP ou PHP.

- `MediaroomConnector`: *stateless bean* que gere a ligação aos *web services* SOAP que compõem a API do Microsoft Mediaroom. Este componente encapsula os *client stubs* gerados localmente a partir dos WSDL disponibilizados pela Microsoft para executar as várias acções necessárias à aplicação, como por exemplo receber informações detalhadas de uma *set-top box* através do seu identificador único, ou disparar notificações para um determinado utilizador avisando que se irá dar início a uma sessão em que está inscrito.

- `PUCApplicationConnector`: faz a ligação do PLAY enquanto *3rd party application* ao PUC. Ao contrário do `MediaroomConnector`, que utiliza SOAP para os pedidos/respostas, as chamadas ao PUC são executadas através do *lookup* do endereço JNDI dos *beans* representados pelas interfaces disponibilizadas na API.

- `PUCResourceConnector`: faz a ligação do PLAY enquanto recurso ao PUC. Este módulo é mais detalhado em 4.3.2.

4.3.2 Recurso

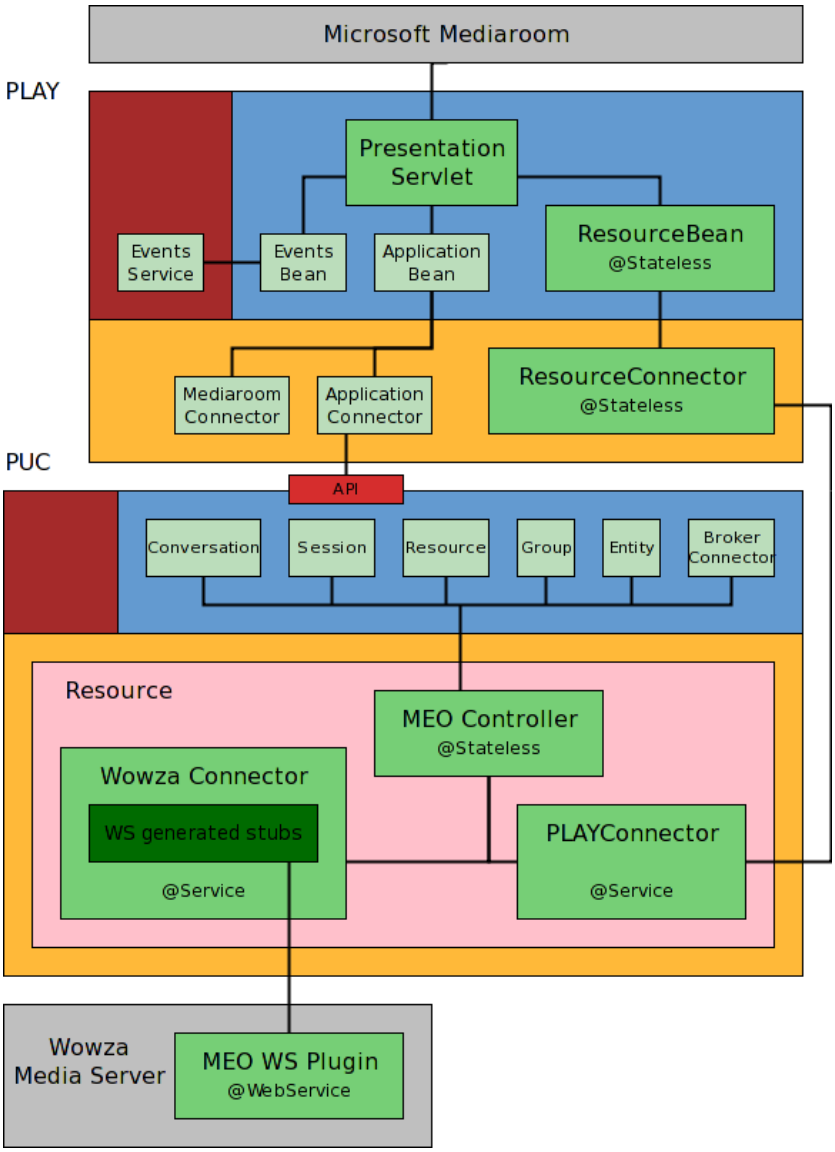

Figura 19: Implementação do PLAY enquanto recurso

A componente de recurso do PLAY é, na verdade, composta por módulos situados em diferentes localizações lógicas.

Ao nível do PLAY estão implementados dois módulos que obedecem à arquitectura em camadas: o `ResourceBean` e o `ResourceConnector`. Como acontece com o restante código, no *stateless bean* encontra-se a lógica de negócio, e no *connector* está implementada a descoberta e utilização da ligação ao `PLAYConnector`. Este componente, localizado já do lado do PUC, é, como todos os restantes *connectors*, o módulo que garante a ligação ao recurso

externo. Embora no caso específico do PLAY este módulo apenas receba informação, em outros recursos implementados a comunicação é nos dois sentidos (como exemplo, o PUC pode informar um servidor de recursos que é necessário reservar uma sala com 10 participantes).

Por cada acção despoletada no PLAY, o `PLAYConnector` informa o `MEOController`, que executa a lógica de negócio implementada e passa a informação para a camada superior (como detalhado em 4.2.3). Ao nível do recurso do PUC existe ainda um outro *connector*, `WowzaConnector`, que gere a ligação ao servidor de recursos Wowza Media Server. Este módulo é chamado quando o utilizador inicia ou termina a partilha do seu vídeo, comunicando ao servidor que deve iniciar/terminar a conversão de um *stream*, qual o URL desse *stream*, e recebendo de volta um novo URL com a localização do *stream* traduzido. Para que o Wowza Media Server possa ser controlado através de uma interface remota, foi necessário desenvolver do seu lado o `MeoWSPlugin`, um módulo que utiliza o sistema de plugins do próprio Wowza Media Server e que implementa um *web service* SOAP, e gerar os respectivos *client stubs* do lado do `WowzaConnector`.

4.4 Interacções entre o PLAY e o PUC

Cada acção invocada pelo utilizador ao nível do PLAY através da sua interface de utilização despoleta a execução de funcionalidades ao nível do PUC. De seguida descrevem-se algumas das interacções entre as duas plataformas.

Início da aplicação (listagem de sessões)

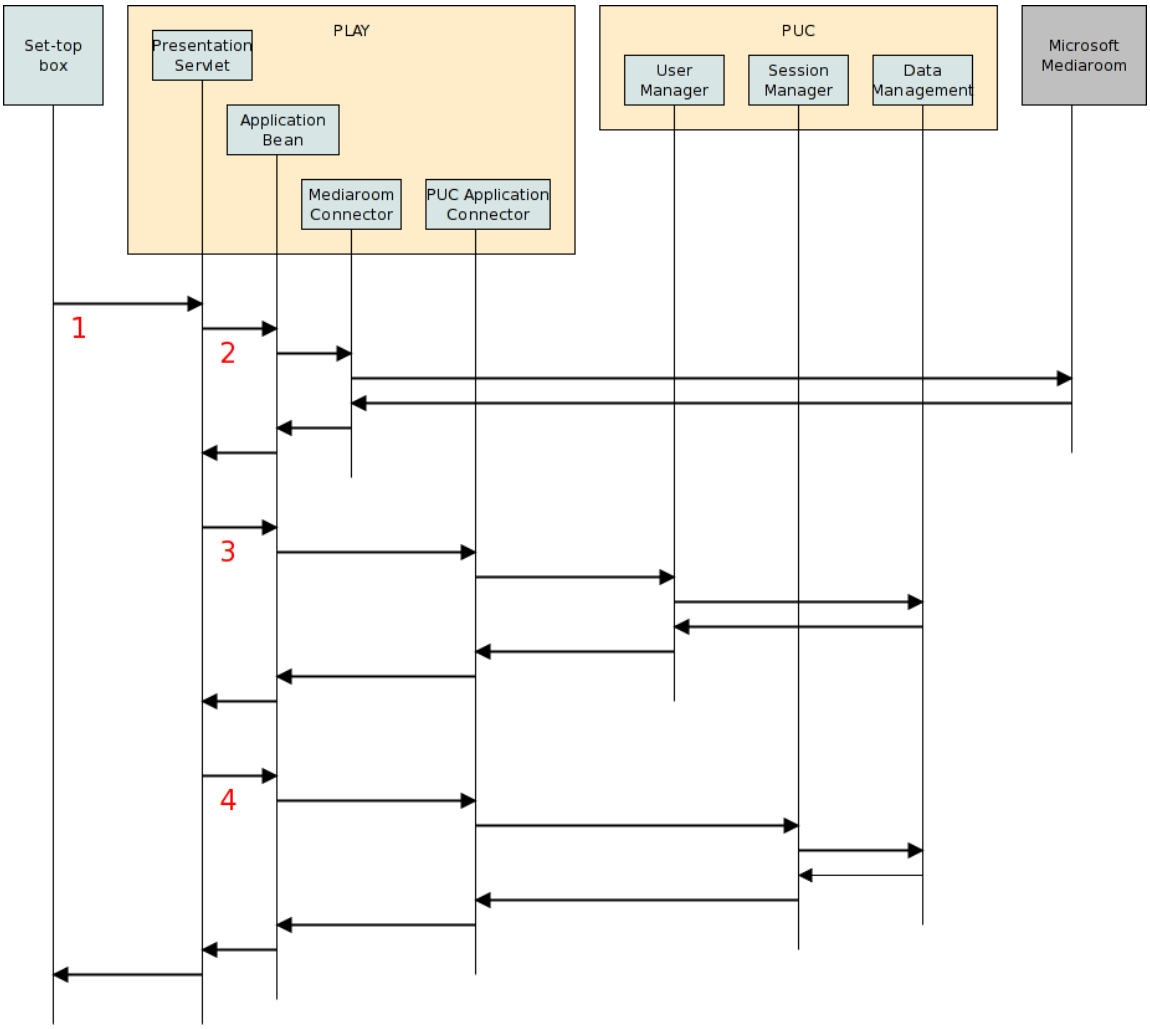

Figura 20: Sequência de acções no início da aplicação

Ao escolher executar a aplicação através do menu da televisão, é feito um pedido HTTP à `PresentationServlet` (1) com o formato:

```
GET presentation?action=start&device=<device_id> HTTP/1.1
```

O identificador único da *set-top box* (`device_id`) é encontrado com recurso a um mecanismo em Javascript. A *servlet* pede então ao `ApplicationBean` para identificar qual o *mediaroom username* do utilizador ao qual pertence a referida *set-top box* (o identificador do utilizador registado na plataforma Mediaroom), através da ligação feita entre os clientes SOAP presentes no `MediaroomConnector` e os *web services* presentes na plataforma

Mediaroom (2). Todos estes dados são guardados numa sessão HTTP exclusiva entre cada utilizador e a *servlet*, de modo a garantir que não é necessário repetir estas interacções durante o resto da execução da aplicação.

Tendo determinado o *mediaroom username,* a *servlet* comunica de novo com o `ApplicationBean` para que este (através da ligação entre o `PUCApplicationConnector` e a API do PUC) peça à plataforma que encontre qual o utilizador do PUC que tem num dos seus vários endereços (*email*, telefone, identificador de serviço de TV) registado o *mediaroom username* que foi encontrado no passo anterior (3). É chamado para isso o `UserManager`, que (com recurso ao `UserService` e aos vários DAOs ligados a entidades do tipo "utilizador"), executa uma *query* à base de dados para encontrar a referência pretendida. No fim deste procedimento determinou-se a identificação do utilizador que está a utilizar o PLAY.

Finalmente, os vários componentes do lado do PLAY comunicam com a API do PUC de modo a encontrar as sessões do PUC em que este utilizador está inscrito (4) – o *manager* utilizado neste caso é o `SessionManager`. Essa lista é apresentada no ecrã da televisão como resposta ao pedido original do utilizador.

Entrada numa sessão

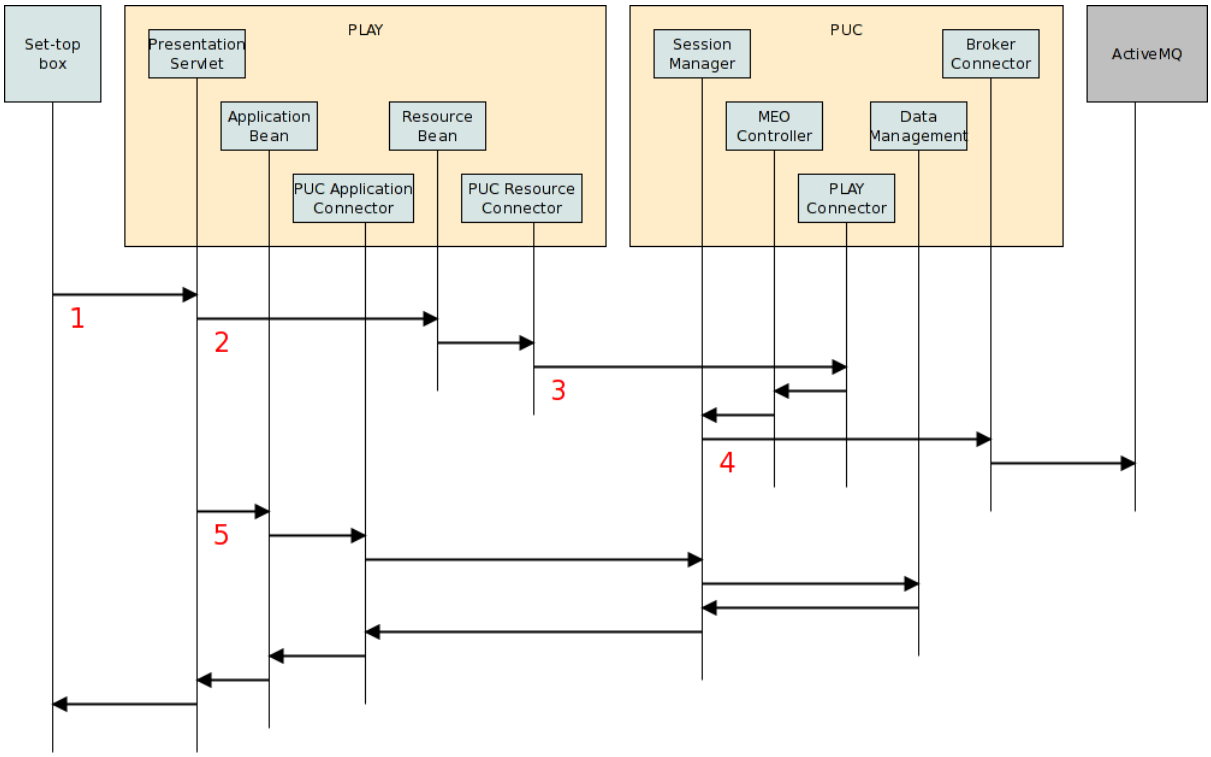

Figura 21: Sequência de acções na entrada numa sessão

O utilizador selecciona com o comando a sessão a que se pretende juntar. Essa acção corresponde a um normal pedido HTTP (1), com o formato:

```
GET presentation?action=join&session=<session_id> HTTP/1.1
```

A `PresentationServlet` tem já a identificação do utilizador guardada na sessão HTTP, não sendo necessário repetir os vários passos de identificação do pedido. Por ser uma acção associada a um recurso, a comunicação é feita desta vez com o `ResourceBean` que, através da ligação entre o `ResourceConnector` e o `PLAYConnectorBean`, informa o `MEOController` qual o utilizador que se juntou a uma sessão, a que sessão se juntou, e com que endereço de rede o fez (2). Esta informação é propagada para o `SessionManager` (3) que, após a execução de alguma lógica de serviço, comunica com o `BrokerConnector` para que este dispare um evento para todas as aplicações informando que o utilizador se juntou à sessão (4).

Do lado do PLAY, a `PresentationServlet` pede então através da ligação à API do PUC a informação necessária acerca da sessão a que o utilizador se juntou: tópico, lista de participantes, e lista de recursos disponíveis (5). Este componente gere ainda a inicialização de um *listener* JMS, `PUCEventListener`, que ficará à escuta de eventos provenientes da aplicação específicos a esta sessão.

Visualização de slideshow

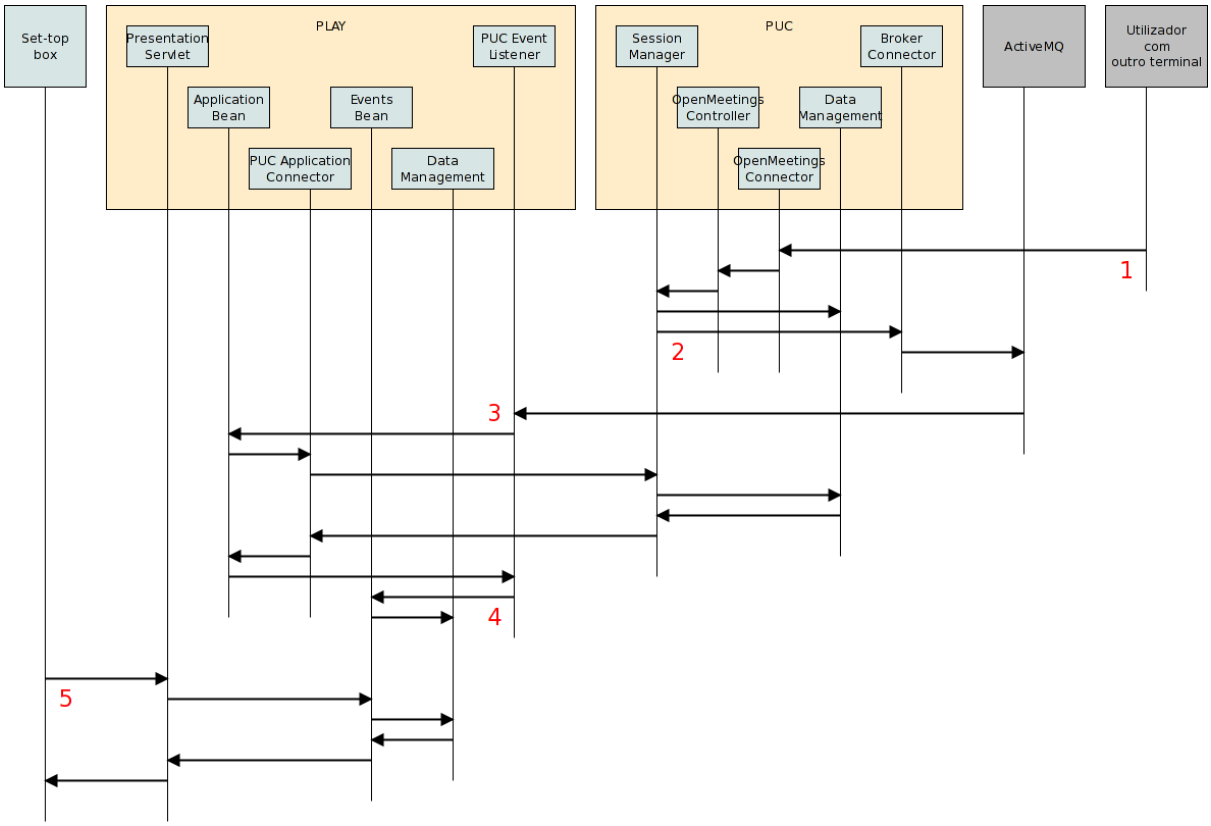

Figura 22: Sequência de acções na visualização de slideshow

A visualização de um *slideshow* através da interface de utilização do PLAY é despoletada quando um utilizador dá início a uma apresentação. A versão actual da plataforma não permite ainda o nível de interacção desejado, pelo que o controlo do apresentação deve ser efectuado através de um outro tipo de terminal. Sendo o terminal IPTV apenas utilizado para a visualização, o sentido da acção de partida é diferente dos exemplos descritos anteriormente: a comunicação é PUC → PLAY ao invés de PLAY → PUC.

Assim sendo, o controlo da apresentação de *slideshows* (início, mudança de *slide*, fim) é feito noutro terminal através de uma interface desenvolvida para o OpenMeetings, um servidor *open source* de sessões colaborativas *web-based* que disponibiliza ao PUC funcionalidades como partilha de vídeo e áudio, *chat,* apresentação d*e slideshows,* partilha de documentos, ou *whiteboard*. A gestão das funcionalidades do OpenMeetings, a sua ligação ao PUC e a implementação de uma interface portável para terminais fixos (computador pessoal) e móveis (telemóvel, *smartphone*) fazem parte do trabalho da tese de mestrado do Hugo Fernandes [37], pelo que não serão descritos em grande detalhe neste documento.

O início de uma apresentação é então assinalado ao PUC sob a forma de um evento de recurso (1). Este evento é recebido pelo *connector* e processado pelo *controller* responsáveis pelo OpenMeetings, o servidor de recursos de onde partiu o evento: `OMConnector` e `OMController`. Estes módulos entregam depois o evento ao `SessionManager` que, após a execução da sua *businness logic*, dispara um novo evento JMS para as aplicações através do `BrokerConnector` (2).

Do lado do PLAY, o `PUCEventListener`, ao receber esse evento, comunica com o PUC (através da habitual sequência `ApplicationBean` e `PUCApplicationConnector`) para receber alguma informação necessária ao processamento do evento, como por exemplo a que sessão está associado o *slideshow* ou qual o utilizador que o despoletou. Essa informação é armazenada na base de dados (através do `EventsBean`) para posterior apresentação na interface de utilização (4).

A actualização da interface de utilizador é um processo que decorre à parte desta lógica. Na verdade, desde a altura em que o utilizador se junta a uma sessão, é iniciado um mecanismo de *polling* em Javascript que, a cada dois segundos (valor configurável), faz um pedido de actualizações à `PresentationServlet` para receber novas informações relevantes à sessão (por exemplo: entrada de novos participantes, início de *slideshows*) (5), com o formato:

```
GET presentation?action=get_updates HTTP/1.1
```

Para cada um destes pedidos, a `PresentationServlet` efectua uma pesquisa na base de dados (através do `EventsBean`) por eventos recebidos desde o último pedido de *updates* deste

utilizador. Os resultados dessa pesquisa passam então pelo *marshalling*, um processo de conversão de objectos Java para uma *stream* XML, que é depois entregue como resposta e processado *client-side* em Javascript. Assim sendo, quando chega ao mecanismo Javascript um processo do tipo `SLIDESHOW_START`, é reservado um espaço na interface para apresentar os slides da apresentação, e o utilizador é avisado de que se irá dar início a uma apresentação.

Durante o resto da visualização, o procedimento é basicamente o mesmo: o apresentador informa o PUC de que houve uma mudança de slide, o PUC comunica com o PLAY a chegada de tal evento, o mecanismo Javascript recebe nos *updates* um evento do tipo `SLIDESHOW_CHANGED` e altera a interface de utilização para apresentar a imagem correspondente ao *slide* actual – o URL a indicar a localização da imagem do *slide* actual é uma das informações que o PLAY recebe do PUC. O mesmo acontece quando termina a apresentação, alterando-se apenas o tipo de evento recebido: `SLIDESHOW_STOP`. O utilizador é informado através da interface que a apresentação teve fim.

Partilha de recurso

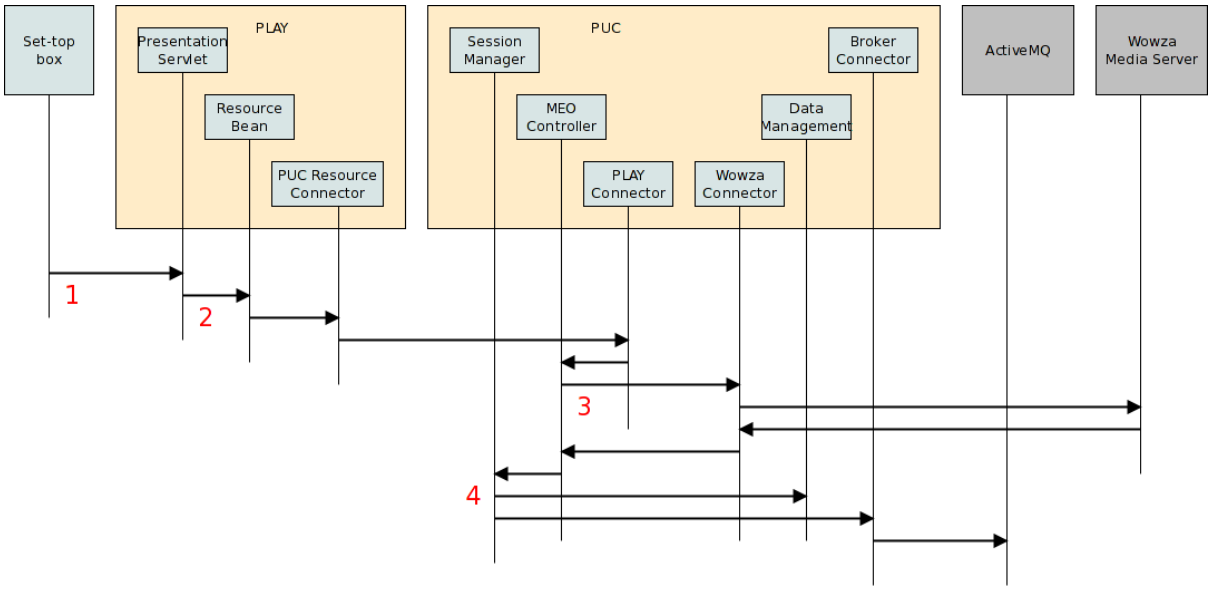

Figura 23: Sequência de acções na partilha de recurso

Uma das funcionalidades implementadas no PLAY é a possibilidade de, caso o utilizador tenha uma câmara de rede integrada na rede doméstica, partilhar o seu vídeo com outros utilizadores.

Quando o utilizador pretende dar início a essa partilha, é enviado um pedido à `PresentationServlet` (1) com o formato:

```
GET presentation?action=toggle_stream HTTP/1.1
```

Na verdade, o pedido é igual tanto para o início como para o fim da transmissão do vídeo. É da responsabilidade da *servlet* recolher as informações necessárias da sessão HTTP e determinar se o utilizador pretende dar início ou terminar a *stream,* através da análise de uma das variáveis lá colocadas.

A `PresentationServlet` informa então o PUC através da habitual sequência `ResourceBean → PUCResourceConnector → PLAYConnector` (2), de que o utilizador pretende iniciar a transmissão do seu vídeo – nesta comunicação vão dados como o *mediaroom username* e o URL de onde a câmara de rede está a transmitir.

Do lado do PUC, o `MEOController` despoleta no Wowza Media Server (através dos clientes SOAP localizados no `WowzaConnector`) o início da conversão do vídeo do utilizador em RTSP para um conjunto de formatos pré-definidos, recebendo e armazenando os URLs de onde o servidor se encontra a fazer *broadcast* de cada um deles (3). De seguida o evento de recurso é propagado para o `SessionManager` que, após executar a lógica de negócio programada, dispara para o `BrokerConnector` um novo evento de forma a informar as aplicações de que um utilizador começou a partilhar o seu vídeo (4). Caso tivesse sido possível implementar uma solução de visualização de vídeo no terminal IPTV, nesta altura o PLAY receberia esse evento de aplicação e reservaria um espaço na interface onde seria disponibilizado o vídeo do utilizador (utilizando o mesmo tipo de interacção entre a `PresentationServlet` e os mecanismos de *polling* Javascript descritos na secção anterior).

5 Validação e ensaios

Após a conclusão da primeira fase de desenvolvimento das plataformas, deu-se início a um processo de validação das funcionalidades implementadas, testando a sua conformidade com os objectivos propostos, a sua adequação aos dispositivos externos nos quais se pretende que funcionem, e a sua integração enquanto aplicação única num cenário de execução o mais próximo possível dos ambientes reais de funcionamento.

Neste sentido, o capítulo presente serve para, em primeiro lugar, introduzir a interface de utilização da plataforma PLAY, detalhando a forma como é feita a apresentação das várias funcionalidades da aplicação no contexto de um terminal IPTV. De seguida apresenta-se a disposição física da bancada de testes, descrevendo os dispositivos utilizados para a execução dos vários testes e como comunicam entre si. Por fim, sumarizam-se os resultados dos testes de validação efectuados a ambas as plataformas (PUC e PLAY) e incluem-se algumas considerações sobre os valores que foram determinados.

5.1 Interface de utilização

O desenvolvimento de interfaces de utilizador obriga a conhecimentos profundos acerca de várias áreas de estudo, como por exemplo o design e a usabilidade.

O objectivo desta tese não era implementar uma solução de produção pronta para apresentar a um utilizador final, mas sim uma plataforma que permitisse demonstrar que o conceito é válido e que tem potencial para futuras aplicações. Desta forma, a interface de utilizador do PLAY, embora com um aspecto cru, permite demonstrar todas as funcionalidades implementadas ao nível da plataforma.

Quando o utilizador liga a aplicação na televisão, é apresentada uma lista com as sessões em que está inscrito:

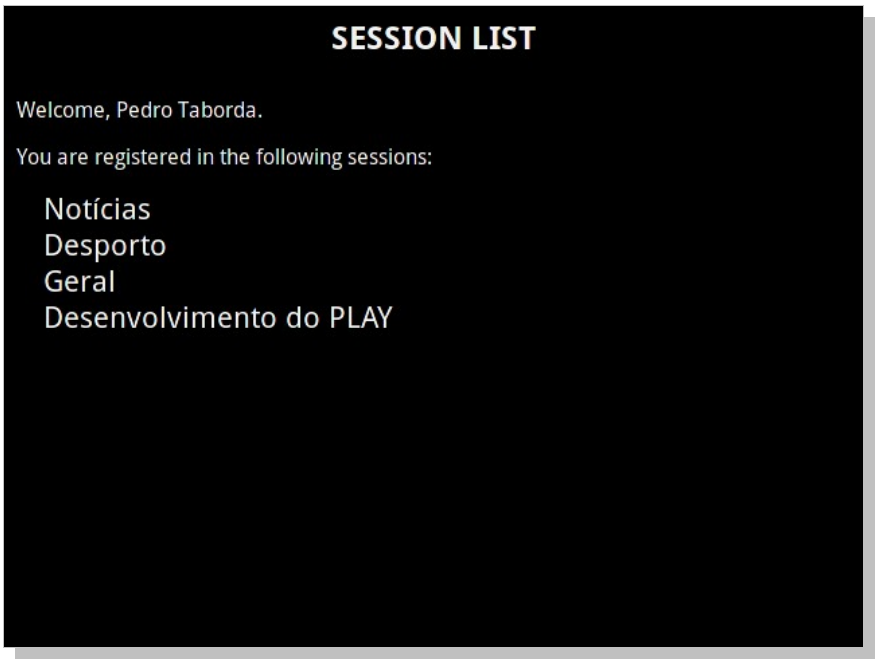

Figura 24: Lista de sessões

A Figura 25 apresenta o aspecto de uma sessão no PLAY. À esquerda encontra-se uma listagem dos participantes actualmente ligados à sessão, com a indicação de com qual terminal o estão a fazer. Nesta figura, por exemplo, podemos ver que estão ligados 4 participantes, encontrando-se 1 ligado através de um computador *desktop* e 3 ligados através de um terminal IPTV.

Na área mais à direita está situado um espaço reservado para disponibilização de conteúdos (neste caso apresentações) e uma lista de acções que o utilizador pode despoletar utilizando o comando da *set-top box*.

Figura 25: Sessão com lista de participantes

Quando o utilizador pressiona o botão verde do comando da *set-top box*, é apresentada uma lista dos recursos disponíveis na sessão. No caso da Figura 26 existem dois ficheiros P*owerpoint* adicionados, e duas apresentações em formato *Flash* criadas automaticamente pelo sistema a partir deles.

Figura 26: Sessão com lista de recursos

Como já foi referido, embora não seja ainda possível através do PLAY controlar a apresentação de um *slideshow*, é possível assistir a uma apresentação que esteja a decorrer. Assim, quando um dos utilizadores ligados à sessão através de um outro terminal (como exemplo, na Figura 27 é o participante Hugo Fernandes) dá inicio à apresentação de um *slideshow,* o espaço reservado para os conteúdos vai sendo alterado, primeiro com um aviso de que irá começar uma apresentação, e depois com os vários *slides* que a compõem.

Figura 27: Sessão com slideshow a decorrer

5.2 Bancada de testes

5.2.1 Disposição

Pretendeu-se criar um ambiente de testes com uma disposição próxima do que se pode encontrar numa solução de produção. Deste modo, tomou-se desde logo a iniciativa de separar fisicamente as comunicações entre os componentes tipicamente presentes numa rede doméstica (*set-top box*, *gateway*) e os que tipicamente se encontram numa rede de serviço (base de dados, servidor de aplicações, *web server*).

O diagrama seguinte representa as ligações entre os vários componentes na bancada de testes:

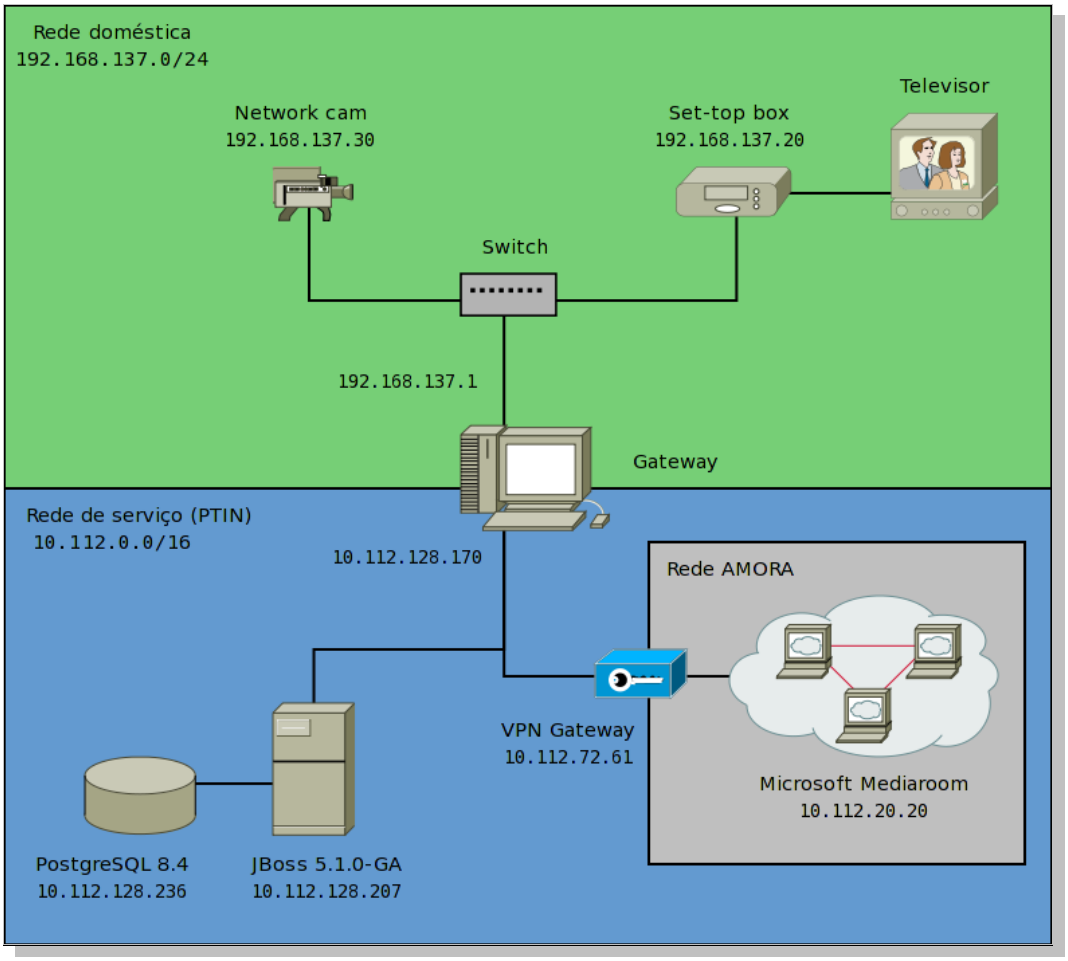

Figura 28: Esquema de rede dos vários componentes da solução

Na rede doméstica encontram-se, como já tinha sido referido, uma *set-top box* (ligada a um televisor) e uma câmara de rede, ambos ligados a um *switch* com um computador *desktop* (configurado para actuar como *gateway*) que liga a rede doméstica (`192.168.137.xxx`) à rede de serviço (`10.112.xxx.xxx`) – de notar que numa instalação em casa de um utilizador, o *switch* e a *gateway* estão habitualmente condensados num único dispositivo. Este *desktop*/*gateway* liga então a *set-top box* ao *application server*, onde estão a executar as duas plataformas descritas neste relatório. Esta máquina tem ainda uma ligação de rede para uma base de dados interna (que serve ambas as plataformas).

Por fazer parte integrante da solução da Microsoft, a *set-top box* precisa de uma ligação constante ao serviço Mediaroom – é dele que recebe não só os conteúdos a que o utilizador pode aceder (sinal de televisão, EPG, notificações, entre outros) mas também importantes informações de sistema (*download* de código para durante o processo de *boot*, por exemplo). Na PT Inovação em Aveiro está em desenvolvimento um sistema piloto que utiliza uma rede privada de distribuição de conteúdos IPTV, o AMORA, onde se encontra uma instalação dos múltiplos servidores que compõem o Microsoft Mediaroom. O acesso à rede AMORA é, como se espera, bastante limitado, sendo o seu acesso efectuado através de um túnel VPN mediante credenciais válidas. Deste modo, torna-se necessário configurar o computador *desktop* para que ligue a rede doméstica (`192.168.137.xxx`) simultaneamente à rede de serviço (`10.112.128.xxx`) à rede AMORA (`10.112.20.xxx`), para que consiga aceder em paralelo aos serviços necessários para o seu funcionamento (nos servidores Mediaroom) e aos serviços da plataforma PLAY (no *application server*).

5.2.2 Equipamentos

A Figura 29 apresenta grande parte dos equipamentos utilizados no desenvolvimento da tese.

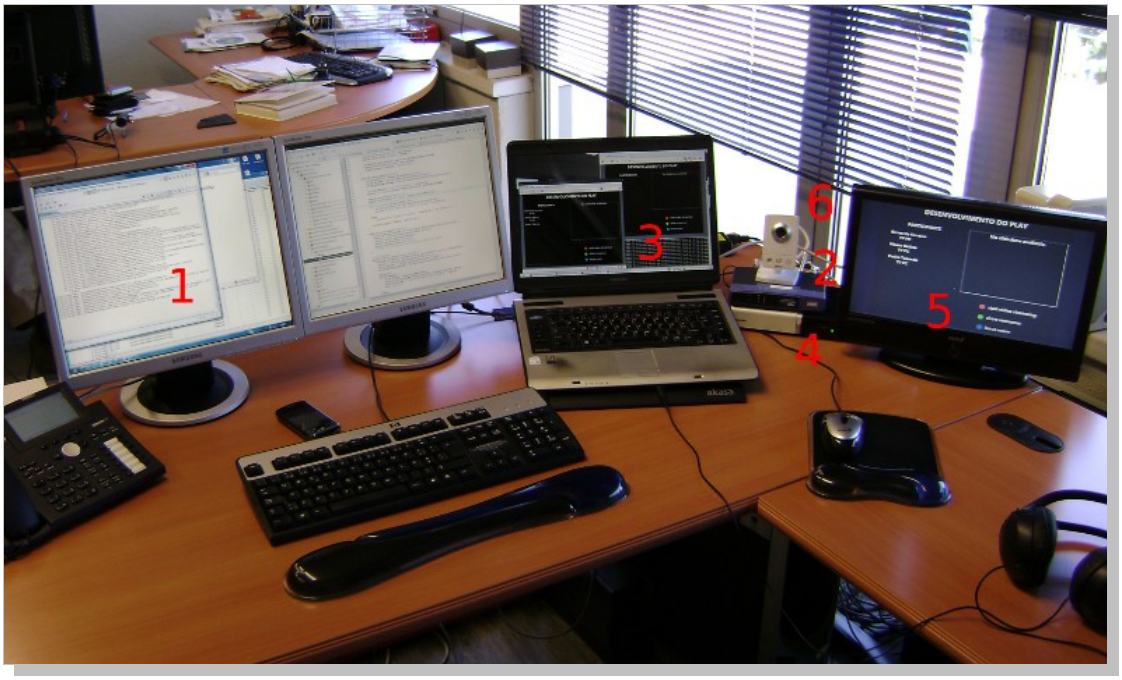

Figura 29: Ambiente de desenvolvimento e bancada de testes

São eles:

1. Computador *desktop* que faz a ligação entre a rede doméstica (192.168.137.0/24) e a rede de serviço (10.112.0.0/16). Neste computador executa também o *application server*, pelo que nesta máquina foram também realizados os testes de validação. Especificações: Intel Core2Duo @ 3.0Ghz, 2.50Gb RAM, Windows 7 32b
2. *Switch* de rede ao qual se ligam os equipamentos da rede doméstica. Este dispositivo, juntamente com o computador *desktop*, simula uma normal *gateway* doméstica.
3. Computador portátil que simula a ligação de outros utilizadores a uma sessão. Nesta máquina foi feita a maior parte do desenvolvimento de ambas plataformas. Especificações: Intel Core2Duo T5600 @ 1.83Ghz, 2Gb RAM, Gentoo Linux (kernel 2.6.34)
4. *Set-top box* Tatung STB 2300 HD [38]
5. Televisão que, ligada à *set-top box* MEO, apresenta a interface de utilização do PLAY.
6. Câmara de rede Axis M1031-W [39]

Além do equipamento apresentado na figura foi ainda utilizado um servidor que armazena a base de dados para os dois projectos, com as seguintes especificações: Intel Xeon @ 2.0Ghz, 4Gb RAM, Ubuntu Server 9.10 (kernel 2.6.28).

5.3 Testes de validação

A execução de testes de validação é um passo essencial no processo de desenvolvimento de qualquer aplicação. As plataformas tratadas neste relatório não escapam à regra, tendo sido determinada uma bateria de testes que serve não só para validar o funcionamento das plataformas em situações de utilização ditas normais, mas também para identificar os problemas da aplicação para que sejam corrigidos nas próximas versões.

As plataformas desenvolvidas, embora estejam intimamente ligadas, têm propósitos e modos de funcionamento bastante diferentes. O PUC, sendo uma plataforma cuja utilização é principalmente programática (os utilizadores acedem às suas funcionalidades através de aplicações), é facilmente testado com recurso a *profilers* e *frameworks* que permitam executar testes de forma programática e exaustiva. Para os testes do PLAY encontraram-se mais dificuldades: uma vez que a plataforma tem uma forte componente de utilização por parte de humanos, não é simples criar *scripts* que simulem fielmente um habitual cenário de utilização. Por outro lado, e como já foi descrito em 4.3, a aplicação é apresentada ao utilizador através de um *browser* localizado na *set-top box*, o qual, por ser um ambiente de execução ao qual não temos acesso directo, dificulta muito as medições dos tempos das interacções.

5.3.1 Criação de utilizadores

A primeira bateria de testes executada no PUC foi a criação e introdução de utilizadores na plataforma. Pretendeu-se medir o tempo que demorava o registo de um conjunto de utilizadores de tamanho variável, e como variavam esses tempos mediante a quantidade de pré-registos já efectuados no sistema. Para simplificar a leitura dos tempos, cada chamada ao PUC introduzia 1000 utilizadores de uma vez.

Os parâmetros de entrada da bateria de testes são:

- número de utilizadores a registar: 1.000, 5.000, 10.000, 50.000. Cada utilizador terá associados 5 `CommAddress` gerados automaticamente: `IPTV`, `EMAIL`, `WAVE`, `FIXED_PHONE`, `MOBILE_PHONE`.
- estado da base de dados: vazia, 5.000, 10.000, 50.000 utilizadores pré-registados.

Como se pode verificar pelo gráfico na Figura 30, o tempo de processamento da operação de inserção de utilizadores aumenta ligeiramente com o aumento do número de utilizadores que se pretende introduzir. Esta diferença pode ser explicada pelo facto de os valores introduzidos serem as médias das 1.000, 5.000 ou 10.000 entradas, pelo que quanto mais perto do fim do teste mais entradas existem já na base de dados. É também interessante verificar que o aumento dos tempos de processamento é pouco notório com poucas entradas na base de dados, mas que que a partir das 10.000 entradas se verifica claramente uma subida mais acentuada desses tempos.

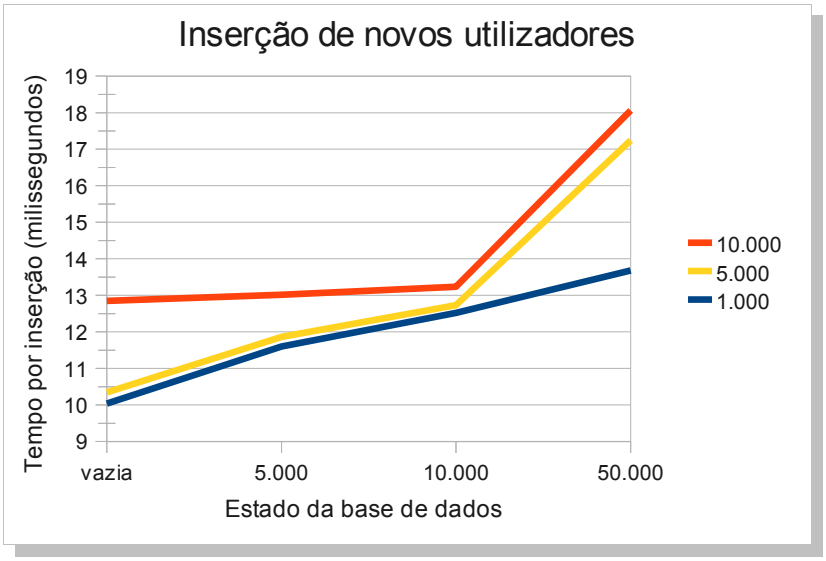

Figura 30: Criação de utilizadores

5.3.2 Criação, inicialização e abertura de sessões

De seguida, foi testado o comportamento da plataforma para a criação, inicialização e abertura de sessões. A linha de execução a seguir é:

1. criação de utilizadores no sistema
2. criação de um grupo de utilizadores
3. criação de uma conversa associada a esse grupo
4. para cada uma das sessões da conversa:

a) criação (*create*) da sessão

b) inicialização (*setup*) da sessão

c) abertura *(open)* da sessão

Os parâmetros da bateria de testes são:

- número de utilizadores registados: 1.000. Destes 1.000 utilizadores registados irão ser escolhidos os participantes em cada sessão, pelo que cada utilizador poderá estar registado em mais que uma sessão simultaneamente. Cada utilizador terá associados 5 `CommAddress` gerados automaticamente: `IPTV`, `EMAIL`, `WAVE`, `FIXED_PHONE`, `MOBILE_PHONE`.
- número de conversas: 5. Uma conversa é apenas um contentor lógico de sessões, o comportamento do sistema é igual para 1000 sessões contidas numa única conversa ou 100 sessões em cada 10 conversas.
- número de sessões por conversa: 10, 20. Uma vez que são sempre criadas 5 conversas, testa-se então a criação de 50, 100 sessões no sistema.
- número de participantes por sessão: 5, 10, 25.

Como se pode verificar nos próximos dois gráficos (Figuras 31 e 32), a quase totalidade do tempo gasto na gestão das sessões é utilizado na sua inicialização e na sua abertura.

A criação da sessão enquanto entidade no sistema, por ser apenas lógica de negócio, revela-se um passo extremamente rápido, sendo até difícil de visualizar nos gráficos. Por necessitarem de comunicação e execução de procedimentos ao nível dos servidores externos de recursos, o *setup* e abertura de cada sessão são procedimentos bastante mais demorados, por razões semelhantes: durante o processo de inicialização é requisitado a cada um dos servidores externo uma pré-reserva dos recursos necessários para a sessão – o que explica o aumento do tempo quando o número de participantes aumenta também; e no processo de

abertura de uma sessão requisita-se a esses mesmos servidores que alterem o estado dessas pré-reservas para "aberto".

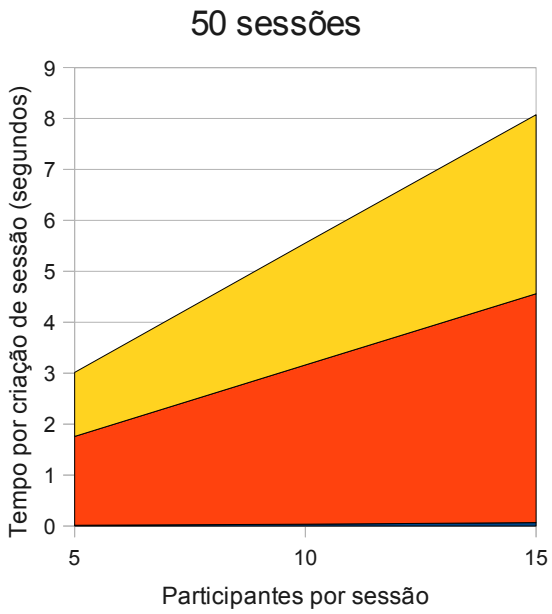

Figura 31: Tempos de processamento na criação de 50 sessões

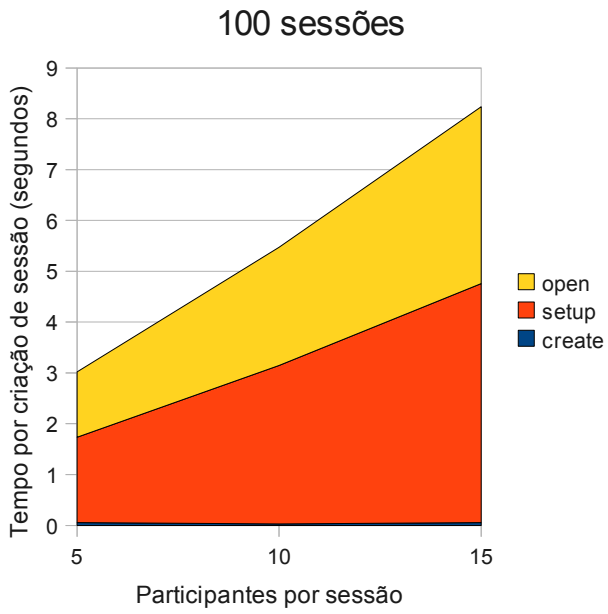

Figura 32: Tempos de processamento na criação de 100 sessões

A análise das próximas figuras, 33 e 34, começa a delinear os responsáveis pela demora no processamento do PUC. Constata-se que entre 70 a 90% do tempo é utilizado ao nível da comunicação com os recursos na altura da inicialização e abertura das sessões, embora o tempo do recurso PLAY seja insignificante. Como já tinha sido introduzido no capítulo 4.3, a implementação do PLAY enquanto recurso serve apenas para a comunicação de eventos de recurso (utilizador junta-se à sessão, início de partilha de vídeo) no sentido PLAY→PUC. Assim sendo, não se aplica a este componente a noção de "reserva de recursos", pelo que o tempo de processamento da aplicação é um valor irrisório.

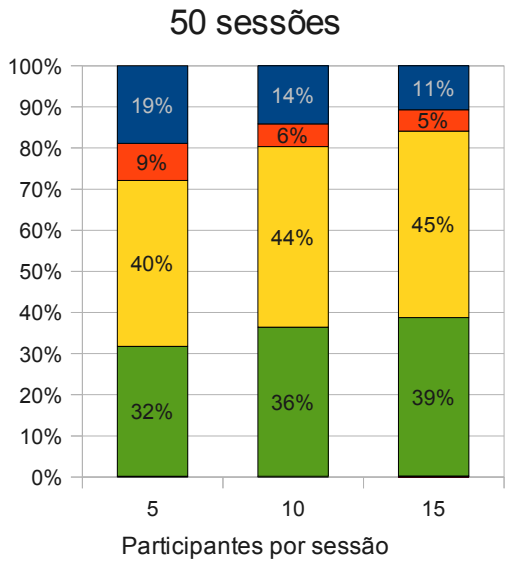

Figura 33: Distribuição dos componentes no PLAY (50 sessões)

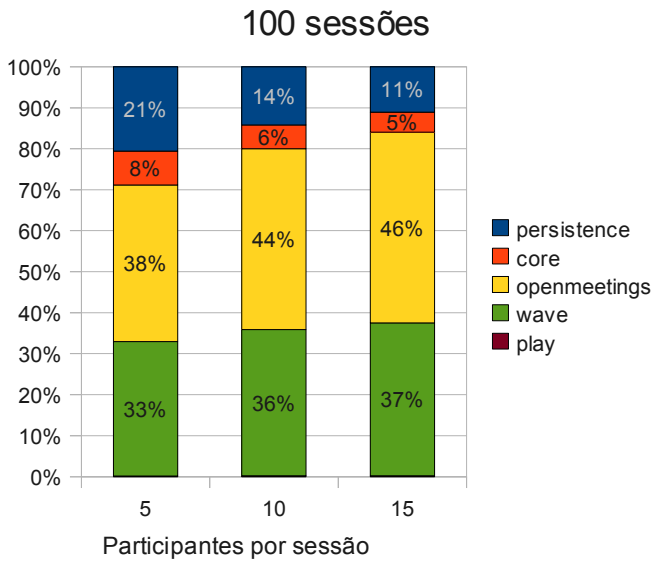

Figura 34: Distribuição dos componentes no PLAY (100 sessões)

Os gráficos nas figuras 35 e 36 corroboram o que já tinha sido introduzido anteriormente. Com efeito, verifica-se que os componentes responsáveis pela comunicação com os servidores de recursos resistem mal ao aumento de entradas no sistema. Uma vez que o aumento do tempo de processamento destes módulos leva necessariamente ao aumento dos tempos totais da plataforma, estes valores deixam no ar a necessidade de fazer algumas mudanças na estratégia de reserva prévia de recursos.

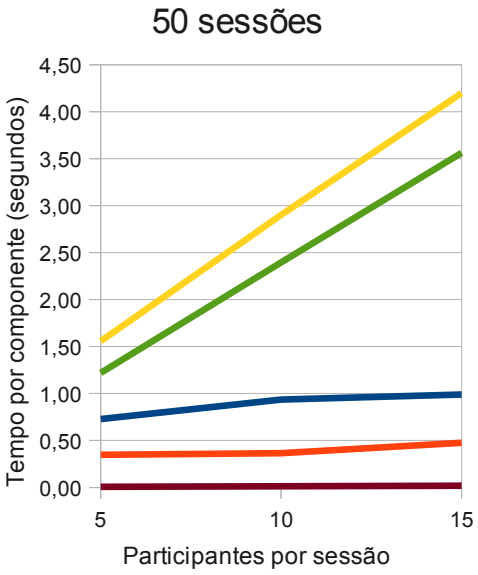

Figura 35: Tempos de processamento por componente (50 sessões)

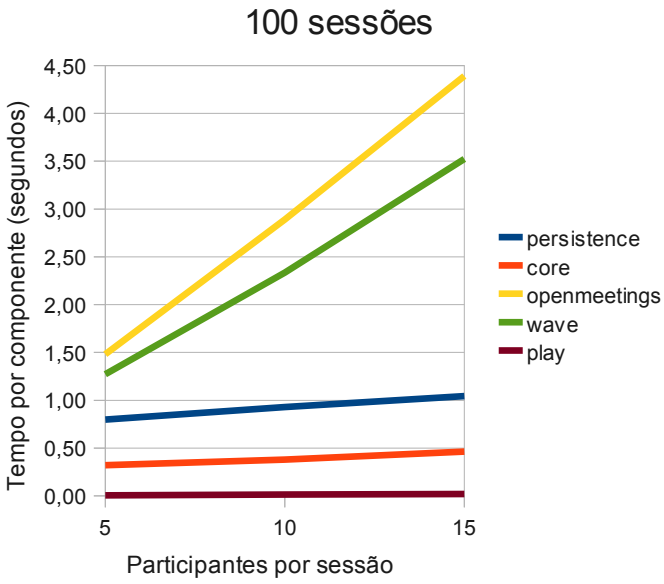

Figura 36: Tempos de processamento por componente (100 sessões)

Testes da versão 1.0

No final da fase de desenvolvimento da versão 1.0 do PUC foram também efectuados testes de validação com o intuito de verificar o desempenho geral da plataforma [22].

Embora não seja completamente correcto fazer uma comparação directa entre os resultados das duas plataformas, uma vez que não foram tomadas as mesmas premissas nem seguidos os mesmos protocolos, a análise dos valores então obtidos ajuda a verificar as melhorias no desempenho da plataforma obtidas com a reestruturação efectuada durante a tese. Adicionalmente, a análise qualitativa das opiniões dos programadores que trabalham directamente no desenvolvimento da plataforma ou em soluções que de alguma forma interagem com ela aponta para reduções substanciais nos tempos de processamento das várias acções, facilitando

5.3.3 Simulação de eventos de recursos

A próxima bateria de testes executada pretendia quantificar a latência do sistema na chegada de eventos provenientes dos vários servidores de recursos. Deste modo, após a inicialização do PUC com alguns valores por omissão, gerou-se um *script* que simulasse periodicamente a geração e envio de eventos de recurso, medindo a demora do processamento da lógica associada a esses eventos através do PUC.

É difícil testar em larga escala no PLAY a produção de muitos eventos de utilização (juntar-se a uma sessão, iniciar a partilha de vídeo), visto que são acções que necessitam de interacção humana. Assim sendo, os testes executados sobre o PLAY foram mais direccionados para a componente de aplicação, focando-se na lógica executada após a recepção dos eventos de aplicação. Uma vez que também se pretendia testar as latências do PUC no processamento dos eventos de recursos, aproveitou-se a oportunidade e testaram-se ambas as plataformas simultaneamente: medindo os tempos ponta-a-ponta, desde o momento que evento é recebido através do *connector* apropriado até ao momento em que termina o processamento do evento do lado do PLAY.

Os parâmetros de entrada da bateria de testes são:

- número de utilizadores registados no PUC: 1.000, cada um com 5 `CommAddress` associados
- número de sessões iniciadas no PUC: 5 conversas, cada uma com 10 sessões, cada uma com 10 participantes
- número de sessões à escuta no PLAY: 5.
- número de eventos de recurso disparados por cada 10 segundos: 5, 10, 20. Estes eventos dividem-se pelas sessões à escuta no PLAY, isto é, os eventos recebidos no PUC são relativos a 5 sessões diferentes. Cada teste demorava 2 minutos, isto é, os dados são relativos a 60, 120 e 240 eventos no total.

A Figura 37 oferece uma visão geral da distribuição dos tempos de processamento medidos durante a simulação de eventos de utilização da aplicação. Analisando o gráfico

facilmente se chega a duas conclusões: em primeiro lugar, a grande maioria do tempo de processamento dos eventos é utilizado ao ao nível do PUC. Em segundo, ambas as plataformas respondem positivamente ao aumento do número de eventos de recurso, não sendo fortemente afectadas por esta variável.

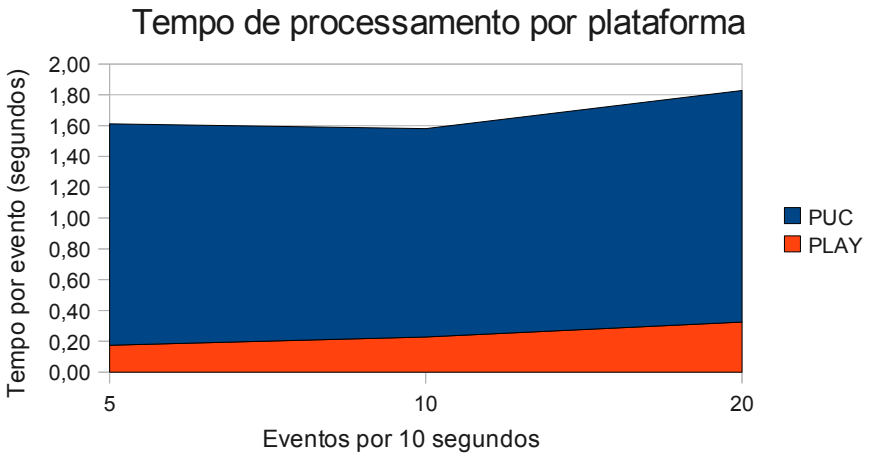

Figura 37: Tempos de processamento por plataforma

O gráfico seguinte (Figura 38) revela que o componente responsável pela persistência dos dados na base de dados ocupa a quase totalidade do tempo gasto pelo PUC no processamento dos eventos. Com efeito, a *businness logic* da plataforma dita que na chegada de um evento de recurso se procedam a bastantes alterações nos dados persistidos anteriormente, pelo que esta percentagem não é de todo surpreendente. Os valores apresentados na Figura 38 representam a média dos valores recolhidos para os vários testes efectuados (5, 10 e 20 eventos por cada 10 segundos).

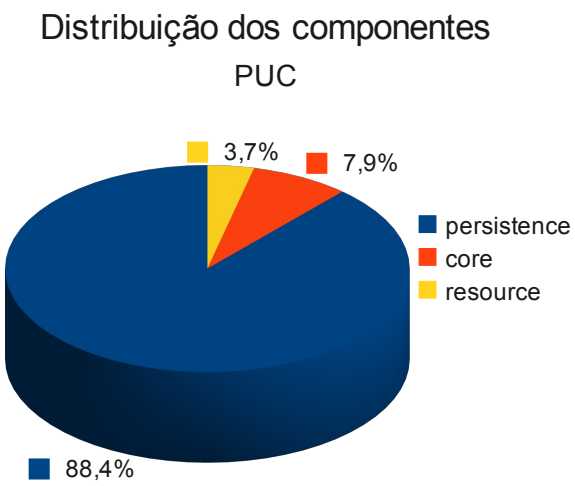

Figura 38: Distribuição dos componentes no PUC

A Figura 39 representa o comportamento de cada componente consoante o número de eventos recebidos por segundo. Verifica-se que o desempenho do componente responsável pela lógica de negócio (*core*) é ligeiramente influenciado pelo aumento da carga de eventos recebidos, sendo mais uma vez evidente que o grande culpado pelo desempenho da plataforma é o módulo que trata da persistência. A diminuição de tempos na passagem dos 5 para 10 eventos não parece fazer grande sentido, pelo que deverá ter origem em alguma carga extra de aplicações externas no computador onde foram executados os testes ou na rede onde este se insere.

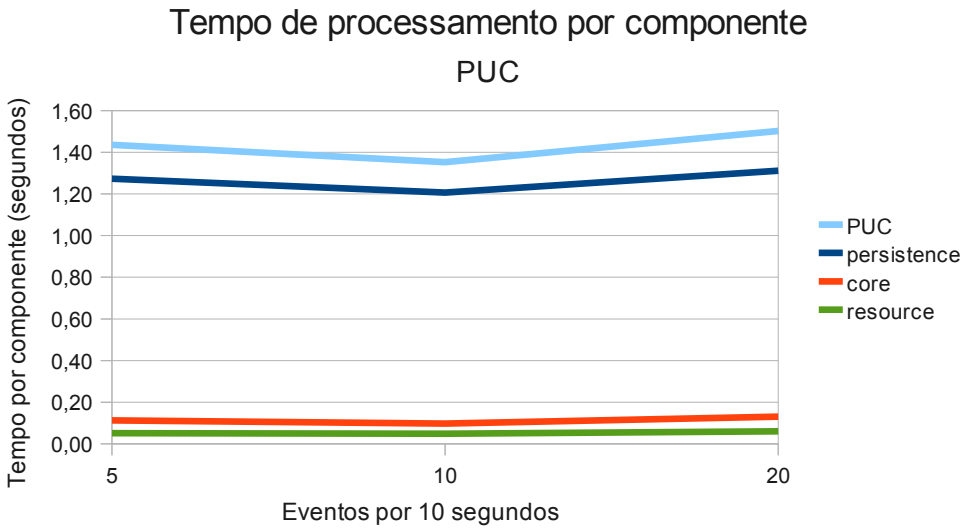

Figura 39: Tempos de processamento de eventos no PUC

Em relação aos tempos medidos no PLAY, convém esclarecer alguns detalhes em relação ao modo como foram efectuadas as medições para que os valores não sejam enganadores.

Como se referiu no início desta secção, executou-se um *script* que, durante 2 minutos, simulou a geração de eventos de recurso para que fossem processados pelo PUC e de seguida enviados para o PLAY. Para que isto aconteça, o PLAY necessita de ter inicializada a sessão a que pertencem os eventos, isto é, necessita de ter pelo menos um cliente ligado para cada sessão em que quer receber os eventos. Como exemplo, imagine-se a seguinte situação: se o *script* simular que estão a ocorrer mudanças de *slides* em duas apresentações que ocorrem nas sessões `Session A` e `Session B`, então o PLAY precisa de ter um utilizador ligado à sessão `Session A` e outro ligado à sessão `Session B` para poder receber esses eventos.

Por outro lado, e como já tinha sido detalhado em 4.3, cada utilizador ligado pela televisão ao PLAY tem a correr um mecanismo de *polling* em Javascript que lhe permite receber os *updates* de informação que ocorrem nas sessões (entrada de um novo participante, início de um *slideshow*, por exemplo). Este mecanismo de *polling* dita que, para cada participante, a cada 2 segundos, seja feito um pedido de novas informações à `PresentationServlet`.

O que isto significa é que cada um dos 5 utilizadores ligados ao PLAY (que possibilitam que a plataforma receba eventos relativos a 5 sessões diferentes) passa também os 2 minutos a fazer de 2 em 2 segundos pedidos de informação de novos eventos. A situação ideal seria separar os tempos de processamento de eventos dos tempos de processamento destes pedidos de informação, o que permitiria a apresentação de dados verdadeiramente relativos ao processamento de eventos. Infelizmente tal não se demonstrou possível, pelo que os tempos de processamento destes pedidos de informação também se encontram incluídos nos dados estatísticos nesta secção. Justifica-se então desta forma a razão do aparecimento da componente `Presentation` (que não tem participação directa no processamento dos eventos) nos gráficos relativos ao PLAY.

Com a visualização do gráfico na Figura 40 constata-se que o processamento na plataforma é dominado pela componente de apresentação. De acordo com os factos introduzidos anteriormente, cada pedido de actualização por parte do mecanismo de *polling* despoleta uma acção de pesquisa na base de dados e um processo de *marshalling* (conversão de objectos Java para *streams* XML) dos dados retornados para envio da resposta. Este processo de conversão é moderadamente pesado para a plataforma, pelo que é natural que a componente de apresentação, mesmo não tendo uma função activa no processamento dos eventos recebidos, ocupe grande parte do processamento da plataforma durante a duração dos testes.

Assim como acontece na Figura 38, os valores apresentados na Figura 40 representam a média dos valores recolhidos para os vários testes efectuados (5, 10 e 20 eventos por cada 10 segundos).

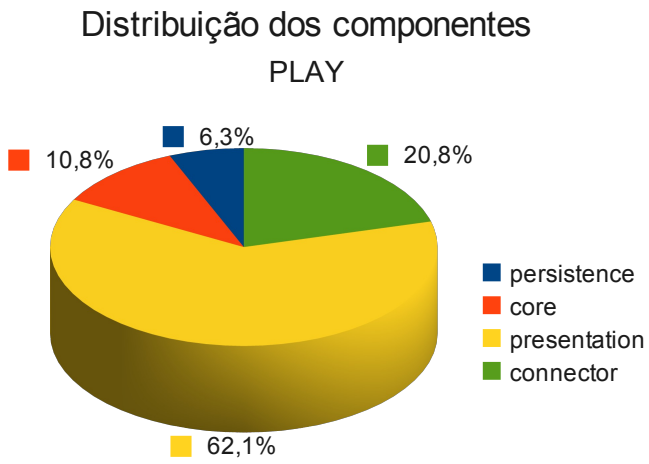

Figura 40: Distribuição dos componentes no PLAY

Ao contrário do que acontece no PUC, onde o aumento do tempo total de processamento deve-se apenas a um único módulo, a Figura 41 revela que no caso do PLAY existem dois principais responsáveis pelo aumento dos tempos totais da plataforma: os módulos `PUCApplicationConnector` (representado dentro do componente *connector*) e `Presentation`.

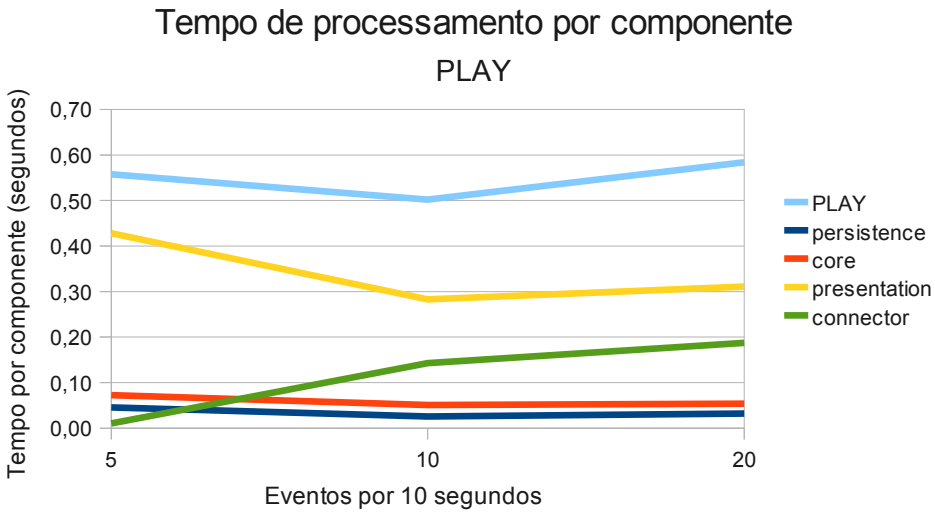

Figura 41: Tempos de processamento por componente no PLAY

Ao receber um evento, o PLAY informa-se junto do PUC (através do `PUCApplicationConnector`) dos detalhes referentes à entidade a que esse evento se refere (se é um novo utilizador, se é um *slideshow,* ou outro). O aumento do número de eventos recebidos influencia a comunicação entre as duas plataformas, e consequentemente o tempo de processamento aumenta também.

O aumento do componente `Presentation`, já explicado anteriormente, mesmo não tendo origem no processamento dos eventos recebidos influencia directamente o tempo total da plataforma.

Assim como aconteceu já com outras medições, e uma vez que os valores recolhidos se encontram na ordem das décimas de segundo, é possível que a diminuição dos tempos na passagem dos 5 para os 10 eventos se deva também a razões externas à plataforma.

6 Considerações finais

6.1 Balanço

O balanço final do trabalho desenvolvido é, na sua generalidade, positivo.

O PUC era até ao início desta tese uma plataforma pesada, complexa e demasiadamente lenta. A reestruturação realizada facilitou o acesso das aplicações aos dados, permitindo uma interacção mais ágil e uma comunicação mais simples. Por outro lado, as melhorias no seu desempenho, não sendo ainda perfeito, deixa no ar a ideia que novos desenvolvimentos poderão colocar a plataforma num nível bastante superior, mais perto de uma versão de produção. Considera-se assim o balanço do desenvolvimento desta componente da tese de mestrado verdadeiramente positivo, uma vez que se cumpriram todos os objectivos propostos durante a fase de planeamento.

O desenvolvimento do PLAY serviu vários propósitos. Considera-se que o principal objectivo, transformar a televisão num terminal de serviços de comunicação colaborativa, foi atingido. A plataforma disponibiliza aos utilizadores de uma forma totalmente integrada com o serviço de IPTV todo um conjunto de informações acerca das várias sessões colaborativas que estão a decorrer e nas quais estão inscritos. Após a entrada numa dessas sessões é possível receber os vários dados referentes a essa mesma sessão: que outros participantes estão ligados e através de que terminais o estão a fazer; que recursos estão partilhados na sala (vídeos, imagens, apresentações). Caso tenha uma câmara de rede associada, é possível através da aplicação partilhar o vídeo dessa mesma câmara para que os outros utilizadores vejam nos seus terminais as imagens desse utilizador. O início da apresentação de um

slideshow e as subsequentes mudanças de *slide* por parte de um outro utilizador no seu terminal são detectadas pela plataforma PLAY e automaticamente disponibilizadas no ecrã da televisão.

No entanto, considera-se que não foi totalmente atingido o nível de interactividade desejado no início da tese. A recepção e envio de vídeo e áudio em tempo real, um dos grandes objectivos propostos, ficou adiada para próximas versões do sistema Microsoft Mediaroom. Embora se tenha conseguido efectuar através do Wowza Media Server a conversão do protocolo nativo de *streaming* suportado pela câmara de rede (RTSP) para o protocolo de *adaptive streaming* suportado na *set-top box* integrada no sistema Microsoft Mediaroom (Microsoft SmoothStreaming), verificaram-se que as latências medidas ponto a ponto tornavam impossível a comunicação em tempo real. O atraso entre a transmissão do vídeo (em RTSP) pela câmara e a recepção no *player* da *set-top box* (em SmoothStreaming) era, com os valores por omissão, na ordem dos 30 segundos. Após a optimização do servidor conseguiu-se reduzir os tempos para valores na ordem dos 10~11 segundos o que, embora seja uma grande diminuição, continua a não permitir uma conversa fluída entre dois ou mais participantes.

De qualquer forma, considera-se que existe ainda espaço para melhorar as funcionalidades implementadas. Na verdade, há bastante margem de manobra para reduzir os tempos medidos nas operações testadas do PUC.

Os resultados dos testes mostram que grande parte do tempo despendido na criação de sessões e participantes é usado ao nível da inserção das entidades na base de dados. Estes valores são justificáveis pelo facto de a plataforma se encontrar ainda numa primeira fase de desenvolvimento. Por haver ainda mudanças ao nível das ligações entre as várias entidades e, consequentemente, das relações entre as muitas tabelas que as representam em disco, a base de dados encontra-se num estado virgem, sem qualquer tipo de optimizações. A introdução de mecanismos de optimização das operações em bases de dados (índices, partições, etc) irá ter com certeza um forte impacto nos tempos de acesso à BD, mitigando alguns dos atrasos medidos durante os testes. No entanto, e como já tinha sido referido, o próprio modelo de dados em que se baseia a solução não se encontra ainda num estado final, pelo que há a possibilidade de vir a sofrer ainda alterações estruturais que permitam uma maior separação

entre as várias entidades e que facilitem as operações ao nível da base de dados. Sendo o desenvolvimento da plataforma um processo cíclico, os testes de validação executados no decorrer desta tese devem ser repetidos no final de todas estas alterações de forma a verificar o real aumento de desempenho.

Verificou-se que outra das principais fontes dos atrasos medidos no sistema é a comunicação com os próprios servidores de recursos. Durante a inicialização de uma sessão, o PUC realiza junto dos servidores de recurso uma reserva prévia dos vários recursos necessários à utilização da plataforma. O *setup* de uma sessão com 100 participantes com suporte a chamadas de vídeo e edição colaborativa de documentos despoleta a criação de uma sala de videoconferência com espaço para 100 participantes ao nível do OpenMeetings e de uma *wave* no servidor Google Wave com espaço para os mesmos 100 participantes. Esta pré-reserva de recursos nos vários servidores externos independentemente da sua utilização real revelou-se um processo extremamente dispendioso, ocupando uma grande porção do tempo total da operação. Convém também lembrar que à medida que se forem adicionando novos servidores de recursos estes tempos irão aumentar de forma correspondente, podendo levar o tempo total de execução da operação a atingir tempos insustentáveis. No futuro deverá ser equacionada a possibilidade de deixar cair a reserva de recursos *a priori* e implementar uma solução de criação de recursos *on-the-fly* a pedido, isto é, só adicionar os participantes aos vários recursos quando efectivamente estes decidirem juntar-se a uma sessão.

6.2 Trabalho futuro

A evolução das duas plataformas descritas neste documento é o rumo natural dos seus processos de desenvolvimento. Existindo na PT Inovação a vontade de aproveitar o trabalho desenvolvido, a correcção dos problemas com que as plataformas se deparam neste momento e, de seguida, a introdução de novas funcionalidades que correspondam aos requisitos dos utilizadores, impõem-se como os próximos passos lógicos a tomar.

Sendo o PUC a plataforma em que se apoiam as várias aplicações (entre as quais se encontra o PLAY), os problemas encontrados durante os testes de validação devem ser os primeiros a ser resolvidos, pois ao resolver os problemas da plataforma de suporte resolvem-se também parte dos problemas que assolam as aplicações. Ao diminuir os tempos de

inicialização e geração dos dados necessários durante a criação de novas conversas, sessões e participantes garante-se uma experiência de utilização mais agradável não só para os utilizadores finais da plataforma mas também para quem a gere.

Só após a resolução dos problemas encontrados e a colocação da plataforma num estado mais estável e robusto se deve apontar o processo de desenvolvimento para a implementação de novas funcionalidades. O *feedback* dos utilizadores será crucial nesta fase, indicando aos responsáveis pelo projecto quais as principais falhas encontradas e quais as funcionalidades que fazem mais falta a quem utiliza a plataforma.

A plataforma PLAY tem potencial para no futuro vir a sofrer grandes alterações. A Microsoft vê no Mediaroom uma solução de sucesso, e tem vindo a aumentar as suas capacidades, integrando-a em novos dispositivos, implementando novas funcionalidades ou suportando novos protocolos de comunicação. Desta forma, espera-se que o PLAY possa acompanhar todas essas novidades.

O PLAY é, nos moldes de hoje em dia, um terminal algo passivo, permitindo principalmente a visualização de dados relativos às sessões e participantes e alguma interacção limitada com os outros utilizadores. No entanto, o único impedimento para a implementação de novas funcionalidades reside do lado do sistema Microsoft Mediaroom. Existe hoje todo um conjunto de funcionalidades que o PLAY poderia implementar caso houvesse suporte do lado do sistema, tornando a plataforma num terminal mais activo.

A integração de um cliente de RTSP nativo na *set-top box* permitirá a visualização de *streams* de áudio e vídeo em tempo real, o que significa a possibilidade de não só partilhar o próprio vídeo com outros (já possível com a versão actual do PLAY) mas também visualizar o vídeo que os outros utilizadores partilham. Deste modo tornar-se-iam possíveis verdadeiras sessões de conferência vídeo e áudio entre múltiplos utilizadores, podendo a televisão complementar ou até mesmo substituir o telefone fixo como dispositivo de comunicação doméstico.

Outra das limitações actuais da plataforma Mediaroom reside ao nível do próprio *software*, que dita a impossibilidade do acesso às portas USB incluídas no *hardware* das *set-top box* por parte das aplicações que nela executam. A abertura destes dispositivos às aplicações significaria não só uma maior liberdade de escolha dos dispositivos de captura de

vídeo, deixando de ser necessário recorrer a *network cameras (*bastante mais dispendiosas para o utilizador), mas também a inclusão de novos tipos de componentes que aumentariam o nível de interacção com os outros utilizadores. Poderia ser possível, por exemplo, ligar um teclado USB à *set-top box* e participar num *chat* através da aplicação, ou ligar uma *pen* ou disco externo e partilhar com os amigos as fotografias e vídeos das férias.

Por outro lado, e como já tinha sido introduzido em 2.2.2.2, existe muito potencial por explorar no modo de implementação da aplicação. A execução da aplicação enquanto conteúdo *web* acessível através do *browser* incluído na *set-top box* é, de acordo com os objectivos propostos no início da tese, bastante aceitável. A escrita deste tipo de aplicações é bastante mais simples e não obriga a um conhecimento profundo da plataforma Mediaroom e das tecnologias que a rodeiam, o que permitiu a rápida implementação de uma interface de utilização que validasse o trabalho desenvolvido no PUC e nas componentes de servidor do PLAY. No entanto, a escolha deste modo de execução traça alguns limites nas funcionalidades disponíveis para a aplicação. Um exercício interessante para o futuro seria a realização de uma avaliação mais profunda destes dois métodos de execução, de forma a executar um *trade-off* entre o tempo e esforço necessários para a aprendizagem de todos os conceitos e tecnologias relativos a cada um deles e o tipo de possibilidades que abrem aos programadores.

7 Bibliografia

[1] "Wowza Media Server" Available: http://www.wowzamedia.com/.
[2] A. Zambelli, "IIS Smooth Streaming Technical Overview," Mar. 2009.
[3] H. Schulzrinne, "RFC 2326 - Real Time Streaming Protocol (RTSP)," Apr. 1998.
[4] "Smooth Streaming : The Official Microsoft IIS Site" Available: http://www.iis.net/download/SmoothStreaming.
[5] ISO - International Organization for Standardization, "ISO/IEC 14496-14:2003: Coding of audio-visual objects -- Part 14: MP4 file format," Mar. 2009.
[6] R.T. Fielding and R.N. Taylor, "Principled design of the modern Web architecture," *Proceedings of the 22nd international conference on Software engineering*, Limerick, Ireland: ACM, 2000, pp. 407-416.
[7] "MEO - O comando é meu" Available: http://www.meo.pt.
[8] "Panasonic VIERA CAST" Available: http://www.panasonic.net/avc/viera/us/viera-cast/index.html.
[9] "SAMSUNG Internet@TV" Available: http://www.samsung.com/pt/consumer/learningresources/medi2.0/internet_introduction.html.
[10] "LG Netcast" Available: http://www.lge.com/us/netcast/index.jsp.
[11] "VIZIO introduces Internet app-enabled XVT HDTV series" Available: http://blogs.zdnet.com/gadgetreviews/?p=5448.
[12] "Sony brings Internet Video TV to the living room" Available: http://news.sel.sony.com/en/press_room/consumer/television/flat_panel_displays/lcd/release/27475.html.
[13] "Yahoo! Connected TV" Available: http://connectedtv.yahoo.com/.
[14] "Skype on your TV" Available: http://www.skype.com/intl/en/allfeatures/tv/.
[15] "Open Network Video Interface Forum" Available: http://www.onvif.org/.
[16] "Physical Security Interoperability Alliance" Available: http://www.psialliance.org/.
[17] "Open IPTV Forum" Available: http://www.openiptvforum.org/.
[18] OpenIPTV Forum, "OpenIPTV Release 1 Specification Volume 1: Overview," 2009.
[19] "Microsoft Mediaroom" Available: http://www.microsoft.com/mediaroom/.
[20] "Microsoft Mediaroom IPTV and Multimedia Platform Debuts at NXTcomm" Available: http://www.microsoft.com/presspass/press/2007/jun07/06-17NXTDebutPR.mspx.
[21] "Vodafone Casa" Available: http://www.vodafone.pt/main/Particulares/vodafonecasa/.
[22] P. Correia, "A Framework for Collaborative Applications," Tese de Mestrado, Departamento de Informática, Faculdade de Ciências e Tecnologia, Univerdade Nova de Lisboa, 2010.

[23] B. Shannon, “JSR 244: Java Platform, Enterprise Edition 5 (Java EE 5) Specification,” May. 2006.
[24] M. Reinhold, “JSR 176: J2SE 5.0 (Tiger) Release Contents,” Sep. 2004.
[25] D. Coward, “JSR 175: A Metadata Facility for the Java Programming Language,” Sep. 2004.
[26] J. Kotamraju, “JSR 224: Java API for XML-Based Web Services (JAX-WS) 2.0,” May. 2006.
[27] A. Mullendore, “JSR 181: Web Services Metadata for the Java Platform,” Jun. 2005.
[28] K. Kawaguchi, “JSR 222: Java Architecture for XML Binding (JAXB) 2.0,” May. 2006.
[29] L. DeMichiel and M. Keith, “JSR 220: Enterprise JavaBeans, Version 3.0,” May. 2006.
[30] L. DeMichiel, “JSR 317: Java Persistence 2.0,” Dec. 2009.
[31] G. King, C. Bauer, M.R. Andersen, E. Bernard, and S. Ebersole, “Hibernate - Relational Persistence for Idiomatic Java,” Apr. 2010.
[32] The PostgreSQL Global Development Group, “PostgreSQL 8.4.4 Documentation.”
[33] N. Deakin, “JSR 914: Java Message Service API,” Dec. 2003.
[34] “Apache ActiveMQ” Available: http://activemq.apache.org/.
[35] B. Ferreira, “Plataforma Unificada e Extensível para Colaboração com Dispositivos Heterogéneos,” Tese de Mestrado, Departamento de Informática, Faculdade de Ciências e Tecnologia, Univerdade Nova de Lisboa, 2010.
[36] P. Delisle, J. Luehe,, and M. Roth, “JSR 152: JavaServer Pages 2.0 Specification,” May. 2006.
[37] H. Fernandes, “Framework Para Aplicações Colaborativas Multi-Plataforma,” Tese de Mestrado, Departamento de Informática, Faculdade de Ciências e Tecnologia, Univerdade Nova de Lisboa, 2010.
[38] “Tatung HD/SD Streaming IP STB : STB 2300” Available: http://www.tatung.tv/default/product/STB2300.html.
[39] “AXIS M1031-W Network Camera” Available: http://www.axis.com/products/cam_m1031w/index.htm.