Faculdade de Engenharia da Universidade do Porto

LiveXtend – Broadcast Events Anytime, Anywhere and in Real-Time

Tiago Alexandre Gonçalves Pereira dos Santos

Relatório do Projecto Realizado no Âmbito do Mestrado Integrado em Engenharia Informática

Orientador: Doutor Pedro Ferreira do Souto

Julho de 2008

© Tiago Alexandre Gonçalves Pereira dos Santos, 2008

LiveXtend – Broadcast Events Anytime, Anywhere and in Real-Time

Tiago Alexandre Gonçalves Pereira dos Santos

Relatório de Projecto realizado no âmbito do Mestrado Integrado em Engenharia Informática

Aprovado em provas públicas pelo Júri:

Presidente: Doutor João Correia Lopes

Arguente: Doutor Rui Moreira

Vogal: Doutor Pedro Ferreira do Souto

15 de Julho de 2008

Resumo

O crescente aumento do número de dispositivos móveis em todo o mundo, aliado ao melhoramento das suas capacidades, tem tornado o mercado do desenvolvimento de *software* para dispositivos móveis bastante atractivo e em expansão.

A somar a isto, está o facto de existir um cada vez maior número de locais cobertos com rede *wireless* e uma continuada tendência para o alargamento das áreas de cobertura, existindo já várias cidades a nacionais e internacionais a oferecerem um serviço gratuito de ligação à Internet por esta via, na área citadina.

O projecto, desenvolvido nas instalações da ClusterMedia Labs, teve como objectivo a criação de um protótipo de um sistema que, aproveitando o potencial oferecido pelos dispositivos móveis actuais, faça a distribuição em tempo real do vídeo capturado por um dispositivo móvel para outros dispositivos móveis.

Assim, foi criada uma solução que permite a um utilizador capturar um vídeo com o seu dispositivo móvel e enviá-lo para outros utilizadores que o poderão visualizar nos seus dispositivos móveis, em tempo real.

No presente relatório encontra-se descrita a especificação do projecto e as decisões tomadas, bem como as características da solução encontrada, nomeadamente a arquitectura e detalhes relevantes da implementação. Encontram-se ainda os testes efectuados e a listagem e análise dos resultados obtidos.

Abstract

The growing number of mobile devices all over the world, together with the improvement of their capacities, has turned the mobile *software* development for mobile devices market very attractive and in expansion.

Adding to this, it is the fact that there is an increasingly bigger number of wireless spots and the tendency for the enlargement of the wireless coverage areas, existing many national and foreign cities providing wireless Internet connection service in all the city's area.

The project, developed at ClusterMedia Labs' facilities, had the purpose of creating a system prototype that, using all the potential of nowadays-mobile devices, distributes in real time a video captured by a mobile device to another mobile device.

Therefore, it was created a solution that allows a user to capture a video with his mobile device and send it live to other users who can visualize it in their mobile devices.

In this report is described the project's specification and the decisions made, as well as the characteristics from the solution found, especially the ones related to architecture and implementation. In this report, there are also listed the tests done to the system as well as the analysis of their results.

Agradecimentos

É-me impossível agradecer neste espaço a todas as pessoas que contribuíram para que conseguisse completar o projecto da melhor forma e que estiveram sempre comigo durante os anos que o curso durou e que antecederam o projecto.

Aos meus pais, irmão e irmã, por tudo.

Ao Nuno, e aos meus tios, que sempre me ajudaram, em especial à Guida que transportou um pouco da minha casa para o Porto.

Aos meus colegas de trabalho, João Oliveira, Luís Certo, Sérgio Lopes e Tiago Araújo, por todo o apoio e amizade.

A todos os meus amigos que estiveram comigo durante o curso.

Ao meu orientador, Prof. Pedro Ferreira do Souto, pela sua disponibilidade e orientações dadas.

Ao Zé Maria, pelas indicações preciosas para a elaboração do trabalho.

Ao Prof. Vidal e ao Prof. Augusto Sousa pela ajuda prestada na resolução de problemas que surgiram durante o projecto.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos e Definições

API	Application Programming Interface
IDE	Integrated Development Environment
CLDC	Connection Limited Device Configuration
Java ME	Java Platform, Micro Edition
JAD	Java Application Descriptor
JAR	Java ARchive
JDK	Java Development Kit
JRE	Java Runtime Environment
JSR	Java Specification Request
JWTK	Java Wireless Toolkit
MIDP	Mobile Information Device Profile
MMAPI	Mobile Media API
MPEG	Moving Picture Experts Group
RAM	Random Access Memory
RTMP	Real Time Messaging Protocol
SDK	Software Development Kit

Aplicação Streamer

Aplicação para um dispositivo móvel que permite capturar um vídeo e enviá-lo por *streaming* para o Servidor.

Receiver

Cliente que recebe um vídeo no dispositivo móvel.

Transcoding ou Transcodificação

Conversão digital directa de um *codec* para outro.

Streamer

Utilizador que captura um vídeo com a câmara do dispositivo móvel e que o envia por *streaming* para o Servidor.

Capítulo 1

Introdução

Este capítulo tem como objectivo contextualizar o projecto, fazendo uma breve descrição da empresa onde foi desenvolvido e explicar qual a motivação e os objectivos propostos para a sua realização. Será ainda feito um breve resumo dos temas abordados nos capítulos seguintes.

1.1 Enquadramento

O presente relatório insere-se no projecto final do curso Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

O projecto teve a duração total de 20 semanas, de 18 de Fevereiro de 2008 a 7 de Julho de 2008 (estando incluído neste período o tempo de elaboração do relatório) e foi desenvolvido nas instalações da empresa ClusterMedia Labs, que se situa na incubadora de empresas da Universidade de Aveiro.

A ClusterMedia Labs é uma *start-up* de base tecnológica, que actua na área de reconhecimento automático de conteúdo audio-visual, assente em duas grandes áreas científicas: processamento digital de sinal áudio e vídeo e inteligência artificial.

1.2 O Projecto

O projecto teve como objectivo o estudo e desenvolvimento de um sistema de envio, distribuição e recepção de vídeo capturado por dispositivos móveis em tempo real.

Pretendeu-se ao longo do projecto estudar quais as possibilidades existentes para a criação dum sistema de distribuição do vídeo capturado pela câmara de um dispositivo móvel, bem como o desenvolvimento de um protótipo desse mesmo sistema.

O sistema a desenvolver deveria estar assente numa arquitectura Cliente-Servidor composta por três módulos principais:

1. Aplicação para um dispositivo móvel que executa a captura do vídeo da câmara do mesmo e o envia por *streaming*, em tempo real, para um servidor.
2. Alteração de um servidor de distribuição de vídeos para poder fazer a recepção de diversos vídeos e a conversão destes para o formato FLV.
3. Aplicação para um dispositivo móvel que permite visualizar o vídeo enviado.

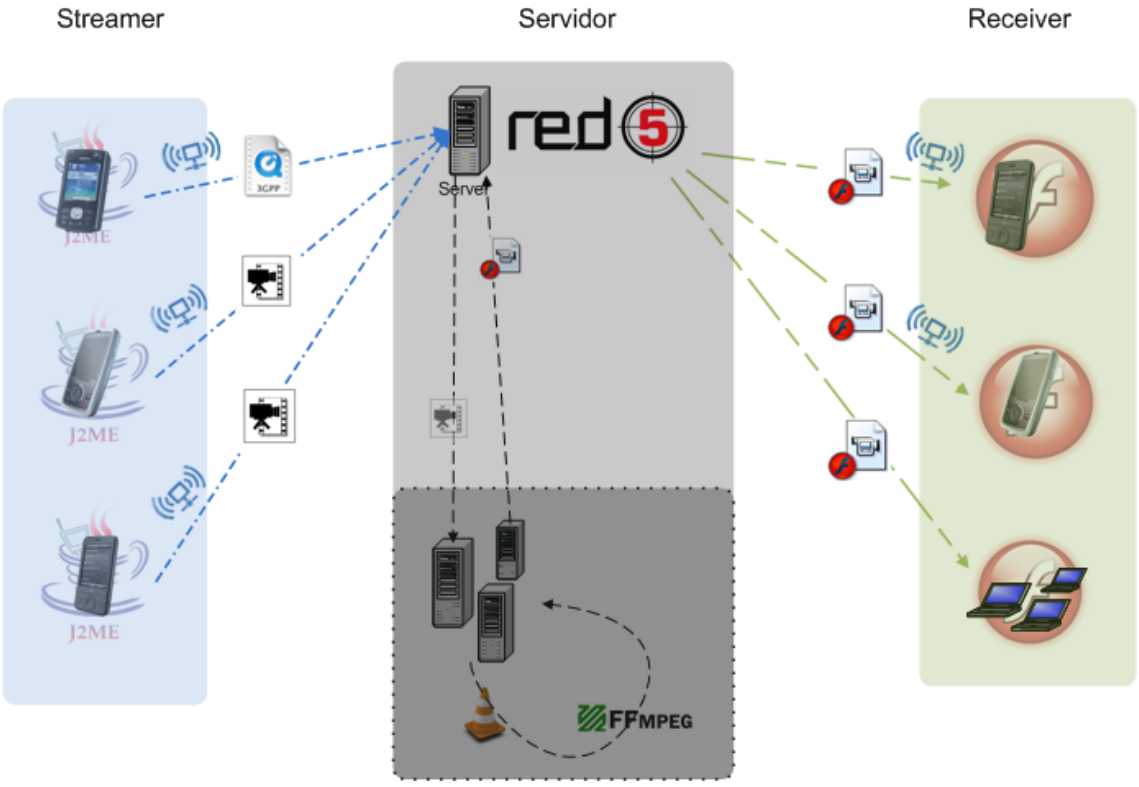

Figura 1 - Vista Geral do Projecto

A Figura 1 mostra uma vista geral do produto a desenvolver. Podem identificar-se os três módulos que a compõem (a azul, o módulo Streamer, a cinzento, o módulo do Servidor, e a verde o módulo do Receiver).

Pode também verificar-se que a Aplicação Streamer correrá sobre um dispositivo móvel que envia um vídeo para o Servidor. Este fará a transcodificação do vídeo, caso necessário e disponibilizá-lo-á para que a Aplicação Receiver possa obtê-lo em tempo real e visualizá-lo.

1.3 Motivação e Objectivos

O mercado dos dispositivos móveis está actualmente em grande expansão existindo, no final de 2007, cerca de 3.3 mil milhões de telemóveis em todo o mundo [Reu07]. À medida que o número de telemóveis cresce, cresce também a percentagem de telemóveis com câmara integrada. Estima-se que tenham sido vendidos cerca de mil milhões de telemóveis no ano de 2007, entre os quais cerca de 115 milhões terão sido *smartphones* [Tom08] (dispositivo móvel com sistema operativo e ao qual estão normalmente associadas características como câmara integrada e placa *wireless* [Dav05] [JoB06] [CEV08]).

À medida que os *smartphones* evoluem, também as câmaras e o seu *software* têm melhorado. Nos próximos anos assistiremos a um aumento significativo da qualidade das câmaras de telemóveis, que poderão mesmo rivalizar com as DSC (Digital Still Camera) [Fei08]. Este facto, aliado às funcionalidades dos *smartphones*, abre novos horizontes que antes não eram possíveis com máquinas digitais compactas e, consequentemente, potencia a existência de novos mercados. Entre estas funcionalidades incluem-se o uso de outros recursos dos dispositivos móveis (GPS, BlueTooth, Wireless, 3G, sistemas operativos com plataformas de desenvolvimento integradas) que permitem, por exemplo, a geo-localização automática das fotografias e a sua partilha imediata, e a existência de *software* que utilize de diversas formas os recursos destas câmaras.

A ideia do projecto surge do aproveitamento destes dois factores: a melhoria das capacidades dos dispositivos móveis e a melhoria das câmaras que estes possuem.

Aliado a estes factores, existe ainda o facto da ClusterMedia Labs desenvolver produtos de análise e catalogação automáticas de conteúdo áudio, que poderão ser aplicados nesta solução de forma a reconhecer de forma inteligente o que está a ser enviado. Apesar de já existirem algumas (poucas) soluções deste género, nenhuma usufrui das vantagens que podem ser retiradas do reconhecimento automático de conteúdos que as soluções da ClusterMedia oferecem.

Importa ainda salientar que existe uma tendência para o alargamento das zonas de cobertura wireless, havendo já várias cidades a estarem totalmente abrangidas por wireless gratuito. Disto são exemplo as cidades de Amarante [Câm08] e São João da Madeira [Câm081]. Desta forma, o recurso à utilização de wireless em detrimento de outros protocolos explorados por operadoras móveis e usualmente pagos (3G, GSM) poderá ajudar a criar um serviço com uma grande área de abrangência e com um preço reduzido para o utilizador.

Por último, o uso de tecnologias recentes e de dispositivos móveis é um factor extra de motivação.

1.4 Estrutura do Relatório

Este relatório é composto por sete capítulos. A primeira – a Introdução – tem como objectivo fazer um pequeno resumo do produto desenvolvido, enquadrá-lo no contexto da empresa onde foi desenvolvido.

No segundo capítulo faz-se uma revisão tecnológica explicando-se as principais características das tecnologias usadas e a forma como se enquadram no problema, apresentando-se também a justificação para as opções efectuadas.

No terceiro capítulo encontra-se a especificação de requisitos do projecto. Aí está explicado o problema e detalhadas as suas necessidades.

No quarto capítulo está explicada e justificada a arquitectura da solução encontrada. O quinto capítulo concentra-se nos detalhes da implementação, mostrando a sequência de funcionamento da aplicação e justificando algumas opções relevantes da implementação.

O sexto capítulo descreve os testes efectuados ao sistema e os resultados obtidos, tentando-se mostrar sempre as conclusões que se podem retirar destes. No sétimo e último capítulo são explicadas algumas conclusões sobre o projecto e o seu desenvolvimento e indicam-se futuras melhorias que poderão ser acrescentadas.

Capitulo 2

Revisão Tecnológica

Este capítulo tem como objectivo descrever algumas das tecnologias usadas e que sejam relevantes para o entendimento da aplicação. Pretende-se ainda justificar as opções tomadas relativamente às tecnologias usadas (mostrando também algumas tecnologias cujo uso foi ponderado), apontando os seus pontos fortes e fracos e os factores que motivaram a sua escolha.

À semelhança da arquitectura do sistema também este capítulo se encontra dividido em três partes principais, sendo tratadas separadamente as tecnologias usadas em cada módulo do sistema. A primeira parte deste capítulo está relacionada com a componente de envio (Aplicação Streamer). A segunda parte listará as tecnologias usadas para encaminhamento e transcodificação dos vídeos, sendo que a terceira parte incidirá sobre as tecnologias para a recepção dos vídeos nos dispositivos móveis.

Depois destas três partes principais, encontra-se um último capítulo onde são brevemente descritas outras tecnologias e ferramentas que foram úteis no desenvolvimento do sistema.

2.1 Aplicação Streamer

A Aplicação Streamer, é o módulo do sistema que corre sobre um dispositivo móvel e que permite capturar e enviar vídeo em tempo real. No capítulo referente à Arquitectura serão dados mais detalhes relativamente a isto.

Começar-se-á por tratar o Google Android, a tecnologia sobre a qual estava inicialmente previsto que servisse de suporte à criação do módulo streamer. Será explicada a motivação para o seu uso e serão também apontadas as razões pelas quais não foi usado. Posteriormente será apresentada uma breve descrição das alternativas surgidas, bem como as razões que levaram a que tivessem sido ou não escolhidas.

2.1.1 Google Android

Inicialmente, estava previsto que o programa a ser desenvolvido para a plataforma móvel fosse assente no novo sistema operativo do Google para telemóveis, o Android. Na altura em que o projecto se iniciou, tinha sido recentemente lançado o Android SDK. Este permite facilitar o desenvolvimento de aplicações para dispositivos móveis que tenham o sistema operativo Android instalado.

O desejo de se desenvolver a aplicação para o Android, apesar de ter sido um requisito inicial, tinha vários motivos:

1- A política interna da ClusterMedia Labs que incentiva o uso de tecnologias recentes e com provável expansão no futuro;

2- A mediatização que tem vindo a existir à volta do Google Android e a crença da ClusterMedia Labs de que este obterá rapidamente uma grande quota de mercado;

3- O facto de o Android ser totalmente Open Source e modificável e que permite um maior conhecimento da arquitectura e um maior controlo sobre o *hardware*;

4- O facto do Android SDK ser numa linguagem familiar – o Java – ainda que com algumas modificações.

2.1.2 O Abandono do Google Android

Ao fim de alguns dias de pesquisa relativa às várias componentes do projecto, (tendo sido dado especial ênfase à Aplicação Streamer), chegou-se à conclusão de que seria inviável desenvolver esta componente da aplicação recorrendo ao Android SDK.

Houve dois factores que levaram a que tal acontecesse:

1. A inexistência de dispositivos com o Google Android instalado "nativamente" (apesar de existirem na altura alguns projectos independentes para a instalação do Android em dispositivos já existentes, não havia garantia do seu correcto funcionamento nem suporte), o que impossibilitava os testes em dispositivos reais;
2. O emulador do Android não suportar captura de vídeo [Goo08].

Estas limitações inviabilizaram o seguimento imediato do desenvolvimento utilizando o Google Android.

Equacionou-se, então, a hipótese de se adiar o desenvolvimento da Aplicação Streamer, iniciando-se o projecto pela alteração do Servidor e do Receiver, na esperança de que entretanto fosse lançado um dispositivo móvel que tivesse o Android como sistema operativo. No entanto, e devido ao facto de não existirem datas oficiais para esse lançamento, a opção de prosseguir com a utilização do Google Android foi abandonada, iniciando-se um estudo para encontrar alternativas possíveis.

2.1.3 Symbian C++ e Open C para Symbian

A primeira alternativa a surgir foi a de utilizar o sistema operativo Symbian e uma linguagem de programação que este suportasse.

A opção de utilizar o Symbian OS estava principalmente baseada no facto deste ter um largo domínio do mercado. No quarto trimestre de 2007, o mercado dos sistemas operativos para *smartphones* repartia-se da seguinte forma: o Symbian OS liderava com 65% da cota de mercado, seguido pelo Microsoft Windows Mobile, com 12%, o RIM BlackBerry com 11%, o iPhone OS com 7% e o Linux com 5% (ver Figura 2) [Can08]. Importa salientar que na altura em que o projecto se iniciou, os dados usados foram os do trimestre anterior, e que o Symbian detinha cerca de 72,5% da quota de mercado.

Existiam na altura três principais versões do Symbian em circulação: o S40, o S60, e o S80. Como já foi referido, seria importante que a aplicação criada fosse desenvolvida para a mais recente tecnologia, neste caso o S80. Contudo, não existiam na altura muitos dispositivos à venda em Portugal com o S80, tendo-se optado pela aquisição dum dispositivo com o S60, o Nokia N80, que tem como sistema operativo o Symbian S60 3rd Edition.

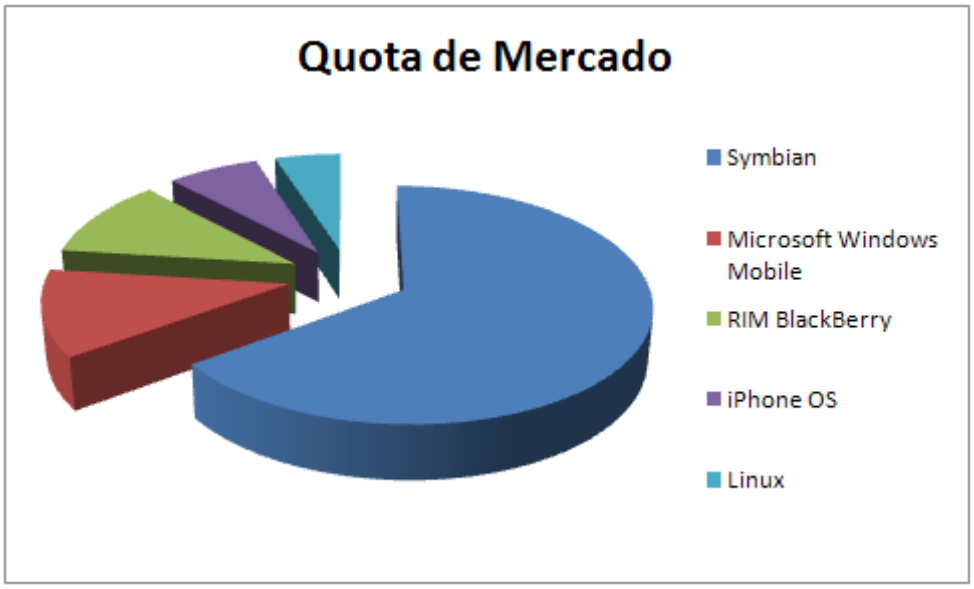

Figura 2 - Quota de Mercado dos SO para Dispositivos Móveis

O Symbian S60 3rd Edition suporta nativamente o Symbian C++ e a instalação do Open C como *plugin*.

Estudou-se assim a hipótese de fazer o desenvolvimento em Symbian C++. No entanto, o tempo de aprendizagem necessário para um conhecimento suficiente do Symbian e com a familiarização com o Symbian C++ eram considerados bastante elevados, e poriam em causa o desenvolvimento do produto no tempo estipulado para tal [Hri051] [The05] [Ant05] [Lar08].

Sobrava, então, a opção de utilizar Open C++. No entanto, também esta opção se revestia de demasiada complexidade para o tempo disponível, sendo mais prático adoptar uma outra linguagem de programação mais simples, como o Java, Python ou Flash Lite [Roc08] [Sym07].

Na altura tomou-se a opção de prosseguir com o Flash Lite, já que se poderia utilizar a implementação do protocolo RTMP (protocolo de comunicação proprietário da Adobe para *streaming* de áudio e vídeo entre um Flash Player e um Servidor) que o Flash Lite traz.

2.1.4 Flash – Kuneri Lite

O Flash Lite vem nativo em muitos dos *smartphones* lançados actualmente (estando previsto que nos próximos 3 anos 46% dos telemóveis sejam lançados com o Flash Lite [Rob08]), possibilita a criação de aplicações visualmente muito ricas, e a sua linguagem – ActionScript – é de aprendizagem rápida.

Ainda assim, o Flash Lite apresenta um problema que poderia condicionar a sua utilização: não consegue aceder a certos recursos do sistema, como é o caso da câmara.

Esta situação era conhecida, tendo sido sugerido logo à partida o estudo de um produto que poderia solucionar esse problema: o Kuneri Lite.

O Kuneri Lite é um Rapid Application Development toolkit (ferramenta para o rápido desenvolvimento de aplicações) que permite, entre outras coisas, integrar em aplicações Flash Lite funcionalidades do sistema operativo, como é o caso do acesso e controlo da câmara [Kun07]. É importante salientar que o Kuneri Lite funciona apenas em sistemas operativos baseados no Symbian S60.

No entanto, também o Kuneri Lite teve que ser abandonado devido a ter limitações no acesso e controlo da câmara que se revelavam incontornáveis e que impossibilitam a criação de uma aplicação de captura e envio de vídeo em tempo real.

Restavam assim duas alternativas: o Java ou o Python. Devido à familiarização com a linguagem e à sua grande portabilidade optou-se pela escolha do Java.

2.1.5 Java ME

A Java Platform, Micro Edition (Java ME) é uma especificação de um subconjunto da Java Platform, que tem como objectivo fornecer um conjunto de APIs para o desenvolvimento de *software* para dispositivos pequenos e com restrições, como é o caso dos telemóveis, e *smartphones* [ASa06].

A tecnologia Java ME abrange ainda assim um grande número de dispositivos com capacidades muito variadas, podendo ir desde dispositivos com 1KB de RAM e com pequenos ecrãs até *smartphones* como o BlackBerry Curve 8320, que tem teclado QWERTY (similar aos dos desktops), ecrãs touchscreens de 2,8 polegadas e 64MB de RAM. [ASa06] [Mob07]

Para que o Java ME possa ser instalado em dispositivos com bastantes limitações mas ao mesmo tempo possa oferecer uma API com mais funcionalidades para dispositivos mais capazes a Java ME foi feita de forma bastante modular. Isto é possível graças à sua divisão em perfis (profiles) e configurações (configurations). [ASa06]

As configurações apontam para um grupo de horizontal de dispositivos (isto é, dispositivos com características semelhantes, principalmente nas restrições de memória). As

configurações são assim as plataformas mínimas que suportam uma gama grande de aparelhos similares. [ASa06]

A plataforma utilizada para o desenvolvimento foi a CLDC (Connection Limited Device Configuration) que tem como alvo dispositivos com memória inferior a 512KB e que é geralmente o padrão suportado pela maioria dos telemóveis. [Nok08]

Os perfis assentam nas configurações (não funcionando sem estas) e são um pouco mais específicos que as configurações, apontando para uma gama de dispositivos mais restrita (dispositivos com interfaces, *hardware* de ligação à Internet e forma de armazenamento de dados semelhantes).

Foi utilizado o perfil MIDP 2.0 (Mobile Information Device Profile) por ser o utilizado no Nokia N80.

A Figura 3 mostra a arquitectura do Java ME, podendo-se nela observar as configurações e os perfis.

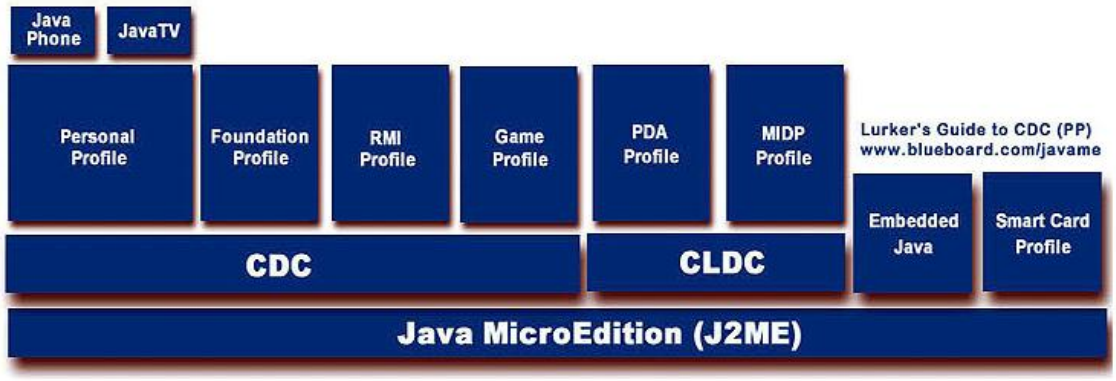

Figura 3 - Arquitectura do Java ME. [ASa06]

O Java ME utiliza uma arquitectura baseada em MIDlets (programa em Java para dispositivos embebidos, e que é normalmente composto por um ficheiro jar (arquivo de Java), que correrá na máquina virtual e por um ficheiro jad (descritor de aplicações de Java), que contém a localização e a descrição dos conteúdos do ficheiro jar). [Wik084]

Os MIDlets têm um funcionamento semelhante a uma máquina de estados, sendo sempre necessário implementar os métodos startApplication (que inicia o MIDlet e o coloca no estado "*running*"), o pauseApplication (que suspende temporariamente a execução do MIDlet, normalmente por ter existido uma acção externa que fez com que o programa tivesse de ser suspendido, como quando o dispositivo móvel recebe um telefonema) e destroyApplication (que termina a execução do MIDlet).

É ainda de salientar que, além da utilização do perfil e correspondente configuração, existem outras "bibliotecas" especificamente desenhadas para o Java ME e que permitem

Figura 4 - Arquitectura do Java ME. [Goy08]

adicionar-lhe funcionalidades extra. Estas bibliotecas têm o nome de JSR (Java Specification Requests) e são descrições actuais de especificações finais e propostas para a plataforma Java. [Jav08]

A arquitectura do Java ME, incluindo JSRs (que corresponde à camada azul) é mostrada na Figura 4.

Uma destas JSR foi particularmente útil no desenvolvimento do produto. A Mobile Media API (MMAPI ou JSR 135) estende as funcionalidades do Java ME fornecendo suporte para áudio e vídeo, já que permite aceder a alguns recursos de dispositivos móveis, como por exemplo a câmara. [Sun08]

Assim, decidiu-se adoptar a tecnologia Java ME para o desenvolvimento da Aplicação Streamer. No entanto, posteriormente foi notado que apesar de ser possível com o Java ME ter acesso à maioria dos recursos de um dispositivo móvel, a sua adequação a aplicações que façam vídeo *streaming* não é a melhor, já que o controlo destes recursos é bastante limitado. [Roc081]

2.2 Servidor

No módulo do Servidor foram usadas três tecnologias principais. A primeira, o Red5, é um Servidor de conteúdos sendo que as restantes, o VLC e o FFmpeg são programas que permitem efectuar a transcodificação de vários formatos de vídeo.

2.2.1 Red5

O Red5 é um Servidor de conteúdos multimédia, desenvolvido em Java sendo Open Source. Tem vindo a ser desenvolvido com o intuito de criar uma alternativa Open Source ao Servidor proprietário da Adobe, o Adobe Flash Media Server.

Entre as principais funcionalidades actuais do Red5 encontram-se o envio *streaming* de áudio e vídeo (FLV e mp3); o suporte do protocolo RTMP; a utilização remota de AMF (ActionScript Message Format), formato este que permite, numa arquitectura Cliente-Servidor, que seja possível evocar métodos remotos.

A arquitectura do Red5 é bastante complexa e não existe nenhum documento formal que a descreva. A versão 0.6 do Red5 tem 831 classes o que, aliado ao facto de não existir nenhum documento de arquitectura torna o seu total entendimento difícil.

Uma das grandes limitações do Red5 é o facto de não ter ainda suportado a recepção de *streams*, fazendo com que apenas possa distribuir vídeos que estejam armazenados localmente.

A ClusterMedia Labs detinha, na altura do início do projecto, uma versão modificada do Red5 que permitia receber um *stream* de entrada e disponibilizá-lo em tempo real. Apesar da solução apresentada ser um pouco rudimentar serviu como base para o desenvolvimento do módulo do Servidor.

2.2.2 FFmpeg

O FFmpeg é uma aplicação que permite efectuar a transcodificação de diversos formatos de vídeo. Tem a facilidade de poder fazê-lo tendo como entrada tanto um ficheiro local como um *stream*. Similarmente, a sua saída poderá ser também um ficheiro ou um *stream*, sendo que neste último caso é usado o protocolo HTTP para a comunicação.

Esta aplicação corre na linha de comandos e utiliza diversas bibliotecas Open Source. Dependendo das bibliotecas que são adicionadas ao FFmpeg aquando da sua compilação, é possível determinar os formatos dos quais e para os quais poderá converter bem como se o conseguirá fazer em *streaming* ou apenas através do ficheiro completo.

A maior utilidade para o projecto provém da conversão de MPEG-1 para o formato FLV (que é o suportado pela versão 0.6 do Red5).

O FFmpeg tem, no entanto, a limitação de não conseguir receber nem enviar *streams* através da Internet, sendo assim necessário utilizar um programa que funcione como um reencaminhador, o VLC.

2.2.3 VLC Media Player

O VLC Media Player é um *player* de vídeo Open Source e que tem diversas funcionalidades, entre as quais a reprodução de diversos formatos e *codecs* de áudio e vídeo, a

correcção de ficheiros corrompidos, a reprodução de conteúdo recebido por *streaming* e o envio de vídeo por *streaming* utilizando diversos protocolos de comunicação, entre os quais o HTTP que é o usado para comunicar com o FFmpeg.

A principal função do VLC no projecto é o de servir de intermediário entre um *stream* recebido pela Internet e o FFmpeg.

2.3 Receiver

O módulo de Receiver que foi desenvolvido é composto por uma aplicação de Flash Lite 3.0.

2.3.1 Flash Lite

O Flash Lite é uma versão reduzida do Adobe Flash Player. O seu objectivo é trazer algumas das funcionalidades do Flash Player para os dispositivos com capacidades mais limitadas que os computadores pessoais. [Ado07]

A versão do Flash Lite utilizada foi a 3.0, que além de permitir, tal como as versões anteriores, visualizar conteúdo multimédia desenvolvido utilizando as ferramentas de desenvolvimento Adobe Flash, permitem também utilizar o protocolo RTMP para *streaming*. [Fla07]

A implementação do protocolo RTMP no Flash Lite 3.0 faz com que seja possível criar rapidamente uma aplicação que envie ou receba um *stream* do Servidor, utilizando para isso a classe NetConnection.

2.4 Outras Tecnologias Usadas

Para o desenvolvimento do projecto foram ainda usadas diversas ferramentas, com diferentes objectivos: umas para auxílio ao desenvolvimento e agilização dos testes, e outras relacionadas com a gestão do projecto. De seguida encontra-se uma breve descrição das ferramentas utilizadas, agrupando-as por categorias.

2.4.1 Ferramentas para Auxílio ao Desenvolvimento

Foram utilizados principalmente dois IDE (Integrated Development Environment) para facilitar o desenvolvimento das aplicações: o Eclipse e o Adobe Flash CS3, tendo o Eclipse sido usado para o desenvolvimento da Aplicação Streamer e do Servidor, e o Adobe Flash CS3 para o desenvolvimento do Receiver.

Eclipse

Todo o desenvolvimento efectuado em Java foi feito utilizando o Eclipse. O Eclipse é um IDE Open Source, desenvolvido em Java, e que disponibiliza muitas das funcionalidades normalmente associadas a um IDE (como editor com *Syntax Highlighting*, compilação

incremental de código, *debugger* e navegador de classes) mas cuja principal funcionalidade se prende com a possibilidade de lhe serem acrescentados novos *plugin*s. [Sco02]

Esta funcionalidade permitiu que lhe fosse facilmente integrado um *plugin* específico para o desenvolvimento de aplicações em Java ME, o Eclipse ME. Este *plugin* simplifica o uso da Java Wireless Toolkit (JWTK, o *toolkit* usado para o desenvolvimento de aplicações em Java ME, com a configuração CLDC e o perfil MIDP [Sun081]), tratando automaticamente e de forma transparente para o utilizador os processos que estão associados à criação de uma aplicação em Java ME. [Ecl05]

A utilização do Eclipse ME permitiu também a integração do emulador do JWTK fazendo assim com que fosse possível testar algumas funcionalidades da aplicação sem ser necessário fazer a instalação destas no telemóvel.

Para o desenvolvimento do Servidor, foi de particular utilidade a integração automática que o Eclipse tem com o Apache Ant (ferramenta Open Source para simplificação do processo de compilação de projectos em Java, e especialmente indicada para projectos grandes [The08]). Visto o Red5 utilizar o Ant para simplificar a sua compilação, a integração do Apache Ant com o Eclipse permitiu que se pudesse testar o Red5 dentro do Eclipse.

Flash CS3 Professional

O Flash CS3 Professional é o mais recente ambiente de desenvolvimento da Adobe para programação em Flash. Foi utilizado para o desenvolvimento da Aplicação Receiver, tendo sido usado em conjunto com o Adobe CS3: Device Central.

O Device Central permitiu acelerar os testes evitando que fosse necessário fazer a instalação da aplicação no telemóvel de cada vez que se quisesse fazer um teste.

2.4.3 Ferramentas de Gestão

Foram utilizadas também algumas ferramentas para a gestão da informação relacionada com o projecto, assim como para a gestão das versões de cada um dos módulos deste.

RedMine e DokuWiki

Foi utilizada uma aplicação Web de gestão de projectos, o RedMine. Esta opção deveu-se sobretudo ao facto do RedMine ser utilizado largamente na ClusterMedia Labs. No RedMine encontrava-se o Logbook do projecto, notas de desenvolvimento, pesquisas relevantes, e era o lugar onde ficavam definidos o plano de trabalho semanal.

Posteriormente, devido a problemas no Servidor, transportou-se esta informação para um outro Servidor, tendo-se utilizado uma *wiki*, tendo-se utilizado para o efeito a DokuWiki.

A gestão das versões foi tratada por um Servidor de SVN da ClusterMedia Labs, tendo-se utilizado dois clientes para o acesso a este: o Tortoise SVN e o *plugin* Subversion, para o Eclipse.

Capítulo 3

Especificação de Requisitos

Este capítulo tem como objectivo listar e detalhar os requisitos associados ao projecto. De uma forma geral, este capítulo explicará em que consiste o projecto e clarifica o que foi acordado com a ClusterMedia Labs ser desenvolvido.

3.1 Visão Geral

O primeiro passo no levantamento de requisitos foi feito de uma forma abrangente, tratando o sistema como um todo, para obter uma visão genérica do que a aplicação final teria que fazer. O diagrama da Figura 5 mostra assim os casos de uso do sistema.

3.1.1 Actores

O sistema tem três actores: o Streamer, que é aquele que deseja capturar um vídeo no seu dispositivo móvel e disponibilizá-lo em tempo real; o Receiver é o utilizador que pretende receber um vídeo a ser partilhado, e o actor Administrador é o responsável pelo funcionamento do sistema. A figura do Administrador é também a do Tester, já que como o sistema a ser desenvolvido é um protótipo, é muito importante que seja possível monitorizar o funcionamento deste.

O Streamer terá como principal caso de uso a captura do vídeo da câmara e o seu envio. O vídeo será posteriormente convertido para FLV, se necessário, e ficará disponível para ser enviado para o Receiver. Assim, o Receiver terá como caso de uso principal a visualização do vídeo enviado pelo Streamer.

O Administrador tem a possibilidade de ver o estado das transmissões dos *streams* enviados pelo Streamer.

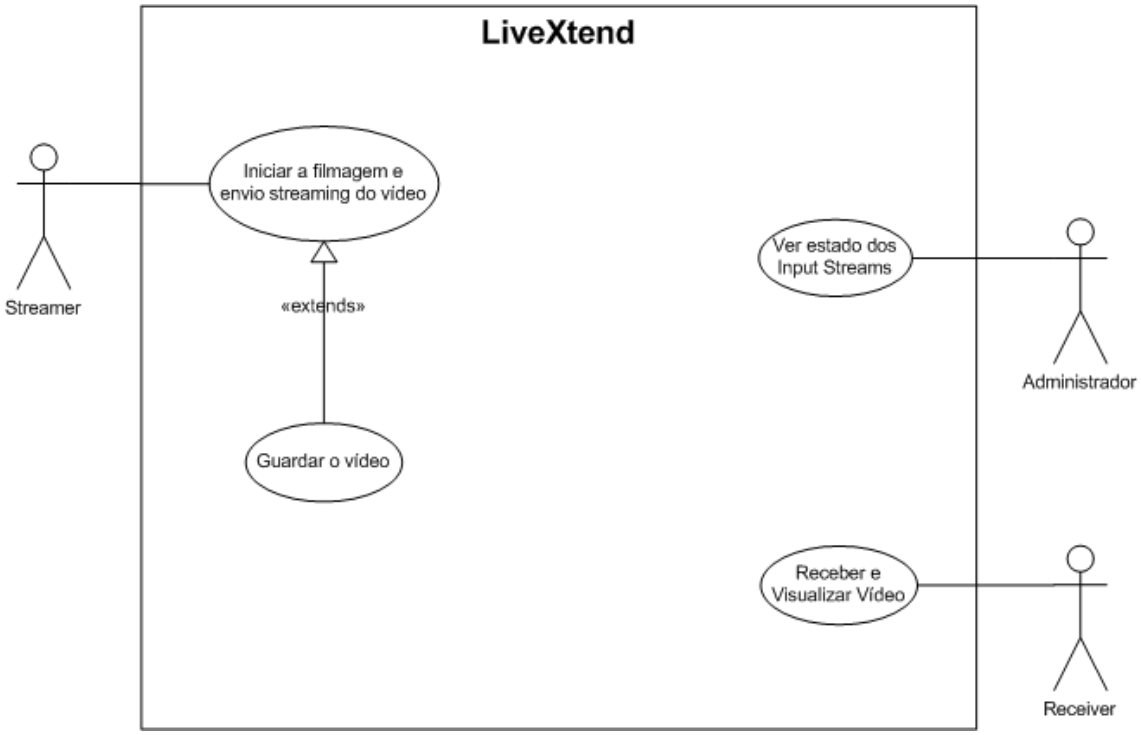

Figura 5 - Casos de uso

3.1.1 Requisitos

Modularidade

Um dos requisitos principais do sistema é que este seja modular, baseado numa arquitectura Cliente-Servidor. Deverá ser composto por três módulos a correrem em plataformas independentes que deverão comunicar entre si via wireless.

O actor Streamer terá acesso uma aplicação (Aplicação Streamer) que funcionará num dispositivo móvel e com a qual poderá capturar e enviar vídeo.

O actor Administrador, deverá ter acesso ao módulo do Servidor. Este módulo estará a correr permanentemente numa máquina (ou várias) que tenha uma boa capacidade de processamento.

O actor Receiver, terá acesso, à semelhança do Streamer, a uma aplicação (Aplicação Receiver) que funcionará num dispositivo móvel independente do do Streamer.

Robustez

O sistema final deverá ser robusto, principalmente no que diz respeito às ligações. O sistema deverá suportar falhas nas comunicações entre os diversos módulos, bem como ter a capacidade de, sempre que possível, retomar transmissões interrompidas.

Baixo *Delay*

Visto ter como objectivo a transmissão de vídeo em tempo-real, é importante que o tempo que decorre entre o início da captura de um vídeo até à sua recepção no dispositivo do Receiver não seja longo.

Apesar de não terem sido definidos valores para este período, foi considerado que qualquer valor abaixo de 4 minutos seria satisfatório para o projecto.

Nos próximos subcapítulos estarão listados e explicados os requisitos referentes a cada um dos módulos.

3.2 Aplicação Streamer

Este subcapítulo descreve os requisitos associados especificamente à Aplicação Streamer, sendo detalhados os casos de uso e apontados outros requisitos importantes

3.2.2 Casos de uso

3.2.2.1 Efectuar Filmagem e Envio do Vídeo

Objectivo

Este caso de uso abrange a principal funcionalidade associada à Aplicação Streamer, a captura do vídeo com a câmara de um dispositivo móvel e o seu envio para um Servidor em tempo real.

Detalhe

A Tabela 1 sintetiza a interacção que o utilizador Streamer terá com a aplicação Streamer, neste caso de uso.

Tabela 1 - Requisitos do Streamer (1)

Descrição da funcionalidade	**Prioridade**
1. Capturar o vídeo de um dispositivo móvel em tempo real e enviá-lo por *streaming* para um Servidor em tempo real.	Alta
2. Suspender a paragem e o envio do vídeo.	Média
3. Escolher o IP do Servidor.	Média

Sequência

O streamer começará por escolher o endereço IP do Servidor onde se deseja conectar. Posteriormente será mostrado outro menu onde aparece no visor o que a câmara está a ver (isto é, não se faz captura do vídeo, o que é mostrado é o que pode ser observado através da câmara). O utilizador deve então premir um botão no menu dando ordem para que seja começada a gravação e o envio dos dados.

Nesse momento poderá, se desejar, parar a captura e a transmissão.

Requisitos de *Software*

A Aplicação Streamer deverá estar assente em Java ME.

A comunicação com o Servidor deverá ser feita utilizando o protocolo HTTP.

Requisitos de Interfaces Externos para o Funcionamento da Aplicação

Para um utilizador poder utilizar a Aplicação Streamer, necessita de um dispositivo que preencha os seguintes requisitos:

- Dispor de câmara (com a qual seja possível capturar vídeo);
- Suportar WLAN (norma IEEE 802.11);
- Encontrar-se num local onde exista uma WLAN que permita aceder ao Servidor (não é imperativo o acesso à Internet, caso o Servidor se encontre na mesma rede);
- Suportar Java ME, com MIDP 1.0 e MMAPI.

Outros requisitos

A comunicação do streamer com o Servidor deverá ser feita utilizando por Wireless, norma IEEE 802.11.

3.2.2.2 Efectuar Filmagem com Envio de Vídeo e Gravação Local

Objectivo

Este caso de uso é semelhante ao anterior, sendo que possibilita também a gravação do vídeo no dispositivo móvel. A sequência de utilização é semelhante à do caso anterior, sendo apenas diferente no momento em que o utilizador dá ordem de gravação, altura em que deverá seleccionar "gravar e enviar".

Também os requisitos são semelhantes, sendo apenas necessário que o Streamer tenha espaço em disco suficiente para guardar o vídeo. Assim, será apenas mostrado o subcapítulo "Detalhe".

Detalhe

A Tabela 2 mostra a interacção que o Streamer tem com a aplicação Streamer, neste caso de uso.

Tabela 2 – Requisitos do Streamer (2)

Descrição da funcionalidade	Prioridade
1. Guardar o vídeo localmente.	Baixa

3.3 Servidor

Este subcapítulo descreve os requisitos associados especificamente ao Servidor, sendo detalhado o caso de uso e apontados outros requisitos.

3.3.1 Casos de Uso

3.3.1.1 Ver Estado dos Input Streams

Objectivo

Este caso de uso descreve a interacção existente entre um Administrador do sistema ou um Tester, e o módulo do Servidor: a visualização do estado dos Input Streams (s*treams* recebidos do Streamer).

Detalhe

A Tabela 3 sintetiza a interacção que o utilizador Administrador (ou Tester) terá com o Servidor.

Tabela 3 - Requisitos do Servidor

Descrição da funcionalidade	Prioridade
1. Visualizar o estado dos Input *Streams* (número de *streams* em simultâneo, tempo há que estão a correr, indicação do IP do streamer, número de pacotes enviados e existência de falhas na ligação).	Alta

3.3.2 Outros Requisitos

Visto o Servidor ser um módulo com pouca interacção com o utilizador, os seus principais requisitos não são descritos através de casos de utilização. Os requisitos restantes estarão listados de seguida.

Objectivo do Servidor

Como já descrito, um dos requisitos do sistema foi que fosse assente na arquitectura Cliente-Servidor. O Servidor tem como objectivo receber os *streams* enviados pelos Streamers, efectuar a seu transcodificação para o formato FLV (caso necessário), e disponibilizar o seu conteúdo para que os Receivers lhes possam aceder.

Deverá também possibilitar a gravação local dos ficheiros recebidos dos Streamers.

A juntar a estas funcionalidades, o Servidor deve possuir um sistema de Logging (registo) que guarde para um ficheiro a informação que é disponibilizada em tempo real para o Administrador e que foi descrito no caso de uso.

Deverá ainda existir independência entre a aplicação responsável pela recepção e envio dos *streams* assim como pela gestão dos ficheiros (Red5 modificado), das aplicações responsáveis pela transcodificação (VLC e FFmpeg). Este requisito tem como objectivo garantir que será possível colocarem-se as tarefas de transcodificação – habitualmente pesadas computacionalmente – noutras máquinas.

Requisitos de *software*

O módulo Servidor é baseado numa versão do Red5 existente na ClusterMedia Labs. Este tinha sido modificado para receber um *stream* e disponibilizá-lo. O principal requisito de *software* foi a utilização desta versão como base para o desenvolvimento do Servidor.

Foi também requisito que, caso fosse necessário efectuar a transcodificação dos ficheiros de entrada, dever-se-ia usar os programas FFmepg e VLC para o efeito.

Foi também requisito a utilização do protocolo HTTP para a recepção dos *streams* enviados pelos Streamers.

Requisitos de Interface Externos

Para ser executado o Servidor, é necessário que a máquina onde fique instalada o Servidor de Red5 modificado, disponha de:

- Apache Ant (versão 1.6 ou superior);
- JDK 1.5
- JRE 1.5

Relativamente às máquinas responsáveis pela transcodificação, deverão ter instalados e configurados na Path os programas:

- VLC
- FFmpeg

As máquinas deverão ainda ser de elevada capacidade de processamento, estando esta necessidade directamente relacionada com o número de transcodificações efectuadas em simultâneo.

Outros Requisitos

Um dos requisitos mais importantes da Aplicação Streamer prende-se com a sua robustez e fiabilidade. O sistema deverá funcionar durante vários dias sem interrupções e deve conseguir tolerar dados de entrada corruptos sem que isso cause alguma falha.

3.4 Receiver

Este subcapítulo descreve os requisitos associados especificamente à Aplicação Receiver, sendo detalhados os casos de uso e apontados outros requisitos relevantes.

3.4.1 Casos de Uso

4.4.1.1 Receber e visualizar vídeo

Objectivo

Este caso de uso abrange a principal funcionalidade associada à Aplicação Receiver, a recepção e visualização num dispositivo móvel de um vídeo no formato FLV.

Detalhe

Na Tabela 4 está sintetizada a interacção existente entre um utilizador e a Aplicação Receiver.

Tabela 4 - Requisitos do Receiver

Descrição da funcionalidade	**Prioridade**
1. Receber e visualizar um vídeo no formato FLV.	Alta
2. Parar o vídeo.	Média
3. Visualizar o tempo de vídeo	Baixo

Sequência

O Receiver terá à sua disposição um pequeno menu onde se pode ver um botão de Play, um de Stop e o tempo de vídeo decorrido.

Ao premir o botão de Play a Aplicação Receiver conectar-se-á ao Servidor e será mostrado no ecrã do dispositivo móvel o vídeo recebido, em tempo real. Caso deseje, poderá também parar a visualização, podendo retomá-la quando desejar.

Requisitos de *Software*

A Aplicação Streamer deverá ser construída para Flash Lite 3.0, para que sejam utilizadas as classes de comunicação para o protocolo RTMP que já traz.

Requisitos de Interfaces Externos para o Funcionamento da Aplicação

Para que um dispositivo móvel seja capaz de utilizar a Aplicação Receiver, precisa de preencher os seguintes requisitos:

- Suportar WLAN (norma IEEE 802.11);

- Encontrar-se num local onde exista uma WLAN que permita aceder ao Servidor (não é imperativo o acesso à Internet, caso o Servidor se encontre na mesma rede);

- Suportar Flash Lite 3.0.

Outros requisitos

A comunicação do streamer com o Servidor deverá ser feita utilizando por *wireless*, norma IEEE 802.11.

Capítulo 4

Arquitectura

Neste capítulo encontra-se detalhada a arquitectura do sistema. É descrita a arquitectura física, mostrando quais os componentes físicos existentes no sistema e as ligações que têm com outros componentes. É também descrita a arquitectura horizontal, mostrando as várias camadas de cada módulo do sistema e explicam-se quais as responsabilidades de cada camada e com que outras comunica.

4.1 Arquitectura física

A Figura 1 mostra o *hardware* e *software* necessário para o funcionamento do sistema, assim como as comunicações existentes entre os componentes.

O sistema tem uma arquitectura do tipo Cliente-Servidor, encontrando-se dividido em três componentes principais: a Aplicação Streamer, o Servidor e a Aplicação Receiver.

A Aplicação Streamer correrá em dispositivos móveis cujas particularidades estão já descritas no Capítulo 3.

A Aplicação Streamer comunica com o módulo do Servidor via *wireless*, utilizando a norma IEEE 802.11 para a ligação wireless e o protocolo de comunicação HTTP.

O Servidor é constituído, tipicamente, por uma máquina que aceita vídeos enviados pelos streamers e os disponibiliza no formato FLV para os Receivers e por, eventualmente, uma ou mais máquinas que farão a transcodificação do vídeo para FLV. Como indicado na Figura 1, a máquina que recebe os pedidos tem a correr o Servidor em Red5.

É possível que as transcodificações dos vídeos recebidos dos Streamers no Servidor sejam efectuadas em máquinas diferentes (sendo esta solução aconselhada em Servidores que tenham necessidade de efectuar a transcodificação de vários vídeos). A área a cinzento-escuro, do módulo do Servidor mostrado na Figura 1 refere-se às máquinas de transcodificação, que poderão então ser diferentes da máquina onde correrá o Red5.

A máquina ou máquinas que fazem a transcodificação, recorrem a dois programas para a conversão de vídeo: o VLC (Video Lan) e o FFmpeg.

Por fim existe ainda um terceiro módulo, o Receiver, que recebe um vídeo enviado pelo Servidor. Este módulo estará a correr num dispositivo móvel cujas características necessárias estão descritas no Capítulo 3.

Este módulo comunica com o Servidor requisitando-lhe um vídeo no formato FLV que este esteja a receber do Streamer, utilizando para isso o protocolo RTMP. A Aplicação Receiver existe sobre a forma de um ficheiro swf, em Flash Lite, que correrá sobre o Flash Lite Player 3.0 que se encontra instalado no dispositivo móvel. No entanto, o Red5 disponibiliza uma aplicação em Flash Player que permite listar e visualizar os vídeos existentes no Servidor. Esta aplicação está disponível numa página Web, também mantida pelo Red5, podendo ser acedida por um Desktop que tenha instalado o Flash Player mais recente.

4.1.1 Necessidades de desempenho

Quanto à capacidade de cada dispositivo, não existem restrições relevantes nem em relação ao dispositivo onde correrá a Aplicação Streamer nem em relação ao que correrá a aplicação de Receiver. À partida, se o dispositivo suportar as plataformas de *software* necessárias (o Java ME e o Flash Lite 3) e as restantes restrições relativas aos "periféricos" que possui (e que estão descritas no Capítulo 3) será capaz de correr as aplicações.

O Servidor, por outro lado, necessitará de ser uma máquina robusta. O Servidor que recebe os pedidos deverá ter uma ligação com boa largura de banda e uma grande capacidade de processamento. A opção de ser utilizado um cluster de computadores seria óptima. Quanto aos computadores que efectuam a transcodificação a capacidade de processamento é ainda maior, sendo a opção de utilização de clusters de computadores a mais indicada.

4.2 Arquitectura horizontal

A arquitectura horizontal mostra as diferentes camadas horizontais do sistema, explicando as relações entre elas.

Cada um dos módulos está dividido em várias camadas, responsáveis por diferentes áreas, como se vê na Figura 6.

A Aplicação Streamer, tem na sua camada superior o interface gráfico, e está suportada por duas camadas inferiores, uma responsável pela captura de vídeo e outra pela comunicação com o Servidor.

O Servidor tem nas suas camadas superiores os módulos responsáveis pela comunicação com os streamers e com os Receivers, que são independentes. Ambos estão apoiados num sistema de gestão de ficheiros. Este sistema consegue tratar dados no formato FLV. Assim, quando os ficheiros recebidos pelo Servidor de *streams* não se encontram nesse formato, é estabelecida a comunicação com um módulo responsável pela transcodificação dos ficheiros recebidos para o formato FLV.

Importa lembrar que as camadas “Servidor de clientes” e “Gestor de ficheiros”, do módulo Servidor encontram-se já implementadas pelo Red5, sendo apenas necessário fazer-lhe alguns ajustes.

A Aplicação Receiver tem uma camada responsável por mostrar o vídeo (a camada de interface gráfico) e uma outra responsável pela comunicação com o Servidor.

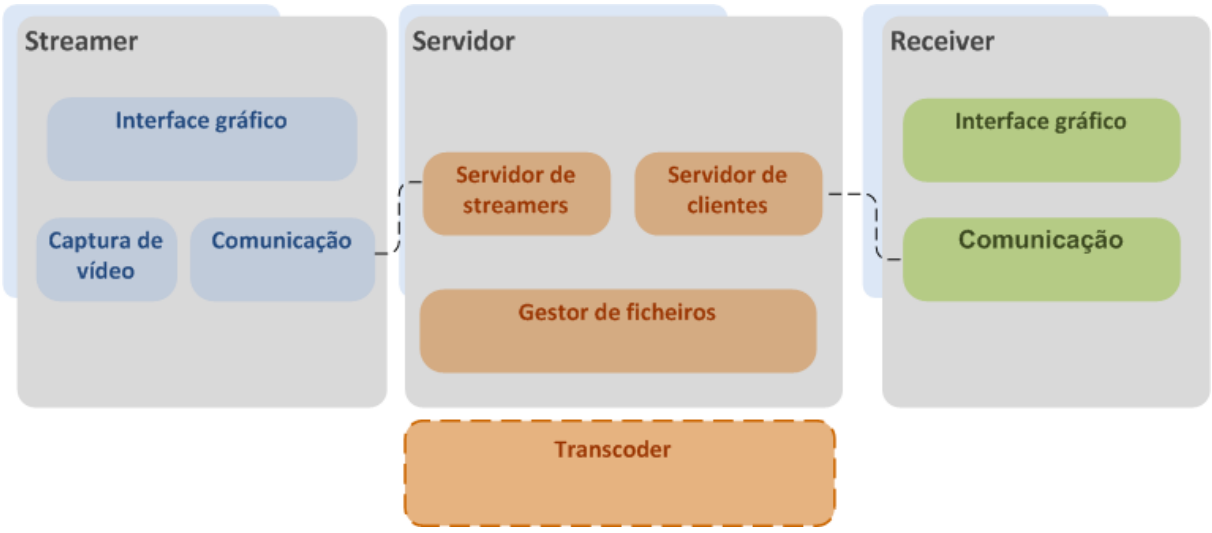

Figura 6 - Arquitectura Horizontal

4.3 Arquitectura vertical

A arquitectura vertical tem como objectivo associar a cada módulo do sistema as funcionalidades e casos de uso que lhe pertencem. A Figura 6 mostra a arquitectura vertical do sistema.

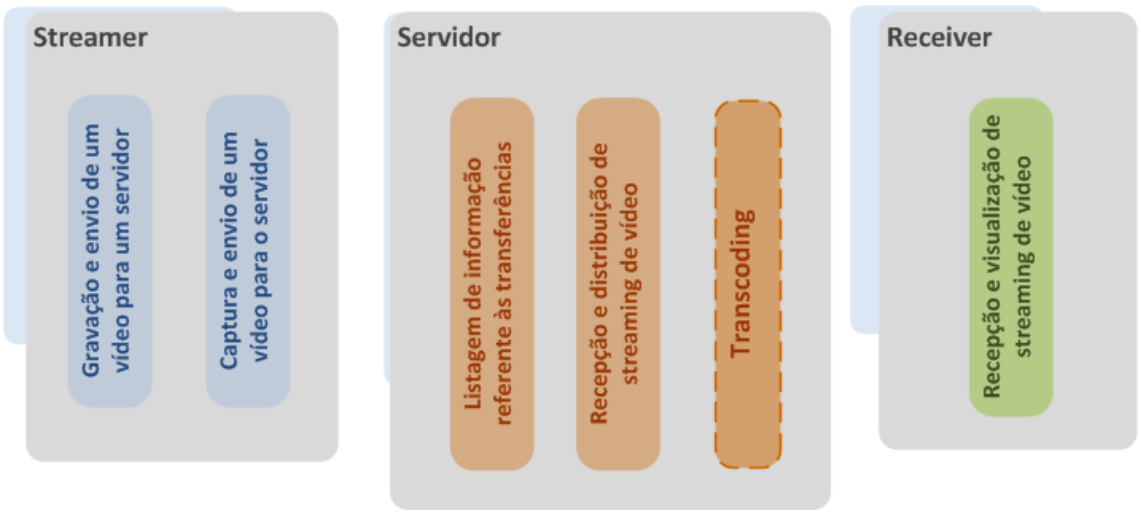

Figura 7 - Arquitectura Vertical

A Aplicação Streamer é a responsável pelos casos de uso relacionados com a produção e envio do *stream* para o Servidor. É neste módulo que um utilizador do sistema poderá efectuar a captura de um vídeo e o seu envio para o Servidor, bem como a gravação do vídeo localmente.

O módulo do Servidor é responsável pela recepção e distribuição dos *streaming*s de vídeo. É também neste módulo que se encontram as funcionalidades relacionadas com a monitorização do funcionamento do sistema, bem como a transcodificação dos vídeos para o formato FLV.

A aplicação Receiver é o módulo que fará a recepção e visualização do vídeo enviado para o sistema pelo Streamer.

4.4 Formatos das Aplicações

Cada um dos módulos é composto por aplicações desenvolvidas em diferentes tecnologias (Java ME, Java e Flash Lite) sendo que cada uma das aplicações terá um formato diferente, e correrá numa plataforma física diferente.

A Aplicação Streamer será composta por um ficheiro .jar (arquivo de Java), com um ficheiro .jad (descritor de aplicações Java) associado e correrá na Java Virtual Machine de um dispositivo móvel que suporte Java ME com o perfil MIDP 1.0 e a MMAPI.

O Servidor Red5 modificado está compilado num ficheiro .jar, que pode ser colocado a correr como serviço ou simplesmente lançado na linha de comandos. Necessita ainda da instalação do programa VLC e do FFmpeg e da sua configuração na Path.

A Aplicação Receiver é composta por um ficheiro .swf em Flash Lite 3.0 que correrá no Flash Lite Player de um dispositivo móvel que o suporte.

Capítulo 5

Implementação

Este capítulo tem como objectivo mostrar o funcionamento e a organização do trabalho desenvolvido. Serão explicados com maior detalhe alguns pormenores importantes da implementação e será mostrada a sequência de funcionamento dos módulos principais, bem como as relações existentes entre as suas classes.

5.1 Abordagem

Neste subcapítulo encontra-se descrita a abordagem feita ao projecto e o plano de trabalhos seguido.

Antes de ser estabelecido um plano de trabalhos e datas para o desenvolvimento de cada módulo, foi feito um levantamento de requisitos e uma definição de alto nível da arquitectura.

Posteriormente foi feito o planeamento para o trabalho futuro. Seriam dedicadas oito semanas para o desenvolvimento da Aplicação Streamer, cinco semanas para a aplicação do Servidor, uma semana para testes de integração e uma breve análise de desempenho, e duas semanas para a aplicação cliente e testes de integração. As três semanas restantes seriam para a escrita do relatório.

Apesar das datas serem flexíveis, o plano foi seguido sem percalços significativos.

5.2 Streamer

Como já referido anteriormente, a Aplicação Streamer foi desenvolvida em J2ME, configuração CLDC e profile MIDP 1.0, tendo sido usada a arquitectura sugerida por Roy, S. visto ser adequada para o problema em questão (utilização de dois formulários: um para a definição do IP do servidor, e outro para visualização, captura e envio do vídeo, bem como *threads* separados para cada uma destas funções) [Sri07].

A sua arquitectura é baseada em MIDlets, existindo uma classe principal (MobileVideoApp) que implementa os métodos relativos aos MIDlets.

No entanto, importa focar este subcapítulo no formulário principal – o VideoRecordingForm – já que é neste que se encontram as funcionalidades principais da Aplicação Streamer: a captura do vídeo (e a eventual gravação em disco) e o envio para o Servidor.

Seguidamente será descrito o processo de captura e envio, identificando-se as classes envolvidas em cada parte do processo, e também detalhes importantes da implementação.

5.2.1 VideoRecordingForm

A classe VideoRecordingForm estende a classe Form do Java ME e é responsável pela disposição dos componentes gráficos no ecrã. Esta classe faz parte da camada de Interface gráfico do Receiver, mostrada no diagrama de Arquitectura Horizontal da Figura 6.

A classe VideoRecordingForm implementa também os interfaces CommandListener, que permite aceder aos botões do dispositivo móvel, e PlayerListener, que permite saber o estado do reprodutor de conteúdos multimédia (e que será melhor explicado no subcapítulo 5.2.2).

Quando é inicializado um objecto da classe VideoRecordingForm, é lançado um objecto da classe CameraThread.

Importa também notar que é nesta classe que são definidos e instalados os *Handlers* responsáveis pelo controlo dos botões do dispositivo móvel e das suas funcionalidades, estando estas associadas ao estado em que se encontra a aplicação, (isto é, se não se estiver a capturar

Figura 8 - Aplicação Streamer a ser Executada no Emulador

um vídeo, o menu relativo aos botões principais do dispositivo móvel mostrará as opções de envio streamer, gravação local, e saída do formulário, como se vê na Figura 8.

5.2.2 CameraThread

A classe CameraThread é responsável por mostrar o que está a ser capturado pela câmara sem que no entanto seja processado. Isto é, a imagem mostrada é como que a de um visor de uma câmara de filmar que mostra no ecrã do dispositivo móvel o que está a ser obtido pela câmara, sem que no entanto seja gravado.

O objecto desta classe cria um *Player* para reprodução de multimédia e associa-o à câmara do dispositivo móvel, para que os dados mostrados pelo Player no ecrã digam respeito ao que está a ser recebido da câmara do dispositivo móvel.

5.2.3 VideoRecordingThread

O objecto da classe VideoRecordingThread é a responsável pela captura e armazenamento (em memória e no sistema de ficheiros, se o caso de uso tratado for o de gravação e *streaming*), do vídeo capturado pela câmara. Esta classe encontra-se na camada de captura de vídeo mostrada no diagrama de Arquitectura Horizontal da Figura 6.

Este Thread correrá em paralelo com o CameraThread e cria um objecto do tipo RecordPlayer que estará associado ao Player criado no CameraThread e fará a captura do vídeo para um Array de Bytes.

5.2.4 *Streaming*Thread

O objecto desta classe é responsável pelo envio do vídeo capturado para o Servidor, em tempo real. A classe *Streaming* Thread encontra-se na camada de comunicação mostrada no diagrama de Arquitectura Horizontal, da Figura 6.

Este Thread é responsável por criar a comunicação com o Servidor, através do protocolo HTTP, enviando o vídeo capturado pelo VideoRecordingThread.O acesso a este vídeo é feito através do uso de uma variável do tipo ByteArrayOutput*Stream* partilhada entre os dois Threads.

5.2.5 Sequência

Nesta subsecção encontra-se explicada a sequência de funcionamento da Aplicação Streamer que é visível na Figura 9. Este diagrama de sequência permite identificar as principais classes envolvidas no processo de captura e envio do vídeo para o Servidor.

Após ser premido pelo utilizador o botão "Gravar e Enviar", são lançados os Threads VideoRecording e o *Streaming*. O Thread VideoRecording inicia a captura da vídeo alojando-o em memória (num ByteArrayOutputStream).

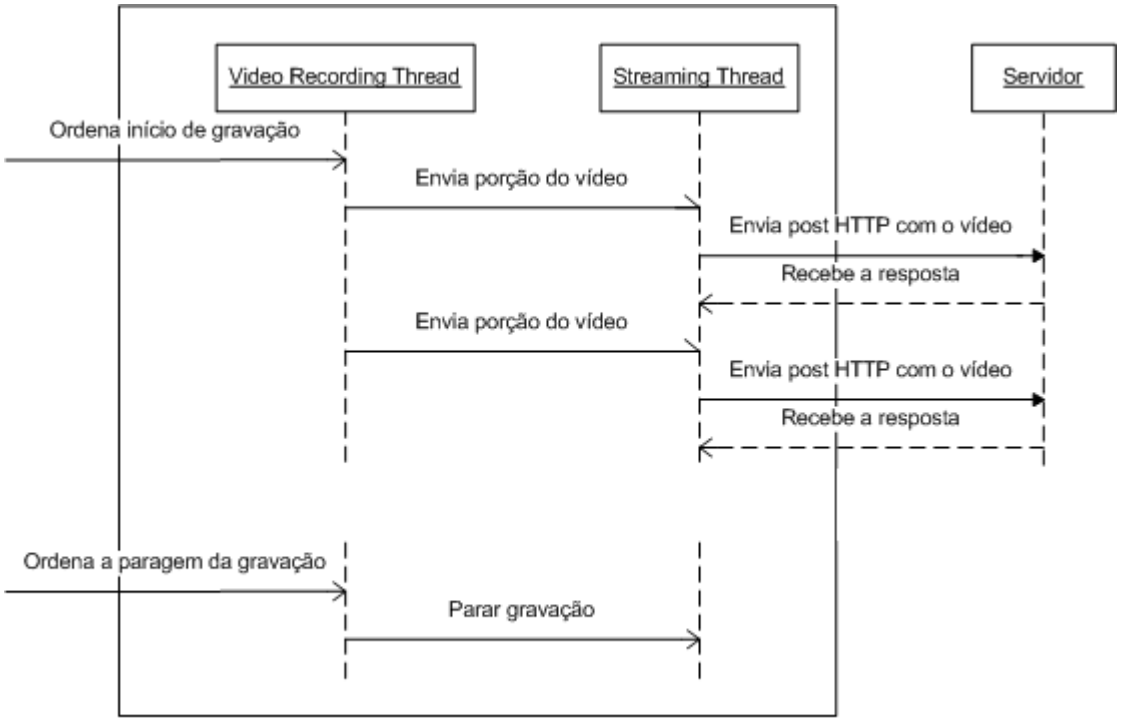

Figura 9 - Diagrama de Sequência do Streamer

Seguidamente, outro *thread* acederia ao conteúdo gravado em simultâneo para o enviar para o servidor, no entanto uma limitação da MMAPI do Java ME fez com que fosse necessário efectuar uma alteração ao planeado. Os métodos de captura de vídeo que esta biblioteca oferece são bastante limitados e não se adequam à criação de uma aplicação para envio de *streaming*.

Para se efectuar uma captura do vídeo da câmara com a MMAPI é necessário chamar dois métodos: startRecord(), que dá ordem à câmara para iniciar a gravação e o commit(), que termina a gravação. Antes de serem chamados estes métodos é necessário definir a variável onde ficará armazenada a gravação (do tipo ByteArrayOutputStream), variável essa que fica inacessível durante o tempo entre a chamada dos dois métodos.

Assim tornou-se impossível aceder ao que estava a ser gravado em tempo real, tendo sido necessário contornar esta situação. A solução encontrada passa pelo chamamento do método commit de tempos a tempos e pela cópia, nessa altura, dos dados lidos da câmara para outra variável do tipo ByteArrayOutputStream. Os dados desta variável podem ser então acedidos pelo *Streaming*Thread, que os enviará para o Servidor.

Assim, ao fim de um tempo (tendo sido usado o valor 8 segundos para os testes) é desligada a câmara e voltada a ser activada para filmar. Entretanto, é comunicado à Thread *Streaming* a existência de uma porção de dados para enviar. Esta comunicação é feita da seguinte forma: quando o *Streaming* Thread não tem mais dados para enviar para o Servidor, testará constantemente o valor de uma variável partilhada entre os dois Threads, e que indica se

já estão disponíveis novos dados para o envio. Caso existam novos dados, o StreamingThread acederá a estes e iniciará a comunicação com o Servidor.

A Thread envia então um post HTTP para o Servidor com o vídeo gravado – um ficheiro no formato 3gp. Quando o Servidor recebe o ficheiro na totalidade, envia para o Streamer a resposta OK, ficando o Thread *streaming* novamente pronto para enviar o próximo ficheiro, caso já tenha sido gravado.

Se entretanto o utilizador decidir parar a gravação e envio para o Servidor, ambos os Threads são interrompidos, não havendo necessidade de comunicar essa situação ao Servidor.

A Figura 10 mostra o estado em que se encontra a Aplicação Streamer após ter sido premido o Stop

Figura 10 - Aplicação Streamer no Nokia N80

Importa ainda assim salientar a solução encontrada para contornar o problema da inacessibilidade dos dados do ByteArrayOutputStream não é ideal, nem mesmo satisfatória para uma eventual aplicação comercial. O tempo gasto enquanto se liga e desliga a câmara é de cerca de 3 segundos, fazendo com que se percam muitas frames e mesmo a linha de continuidade do filme.

5.3 Servidor

Como explicado na descrição das tecnologias, o Red5 é um Servidor de conteúdos Open-Source, feito em Java e que tem 831 classes, na versão 0.6. Este Servidor apenas consegue disponibilizar conteúdos que se encontrem localmente, existindo à data do início do projecto na ClusterMedia uma versão modificada e pouco robusta que permitia aceitar um *stream* de entrada.

Assim, para a implementação foi necessário entender arquitectura do Red5 e alterar e acrescentar algumas classes para que este pudesse receber *streams* e disponibilizá-los em tempo real, ao invés de apenas permitir o acesso a vídeos no formato FLV que se encontram localmente.

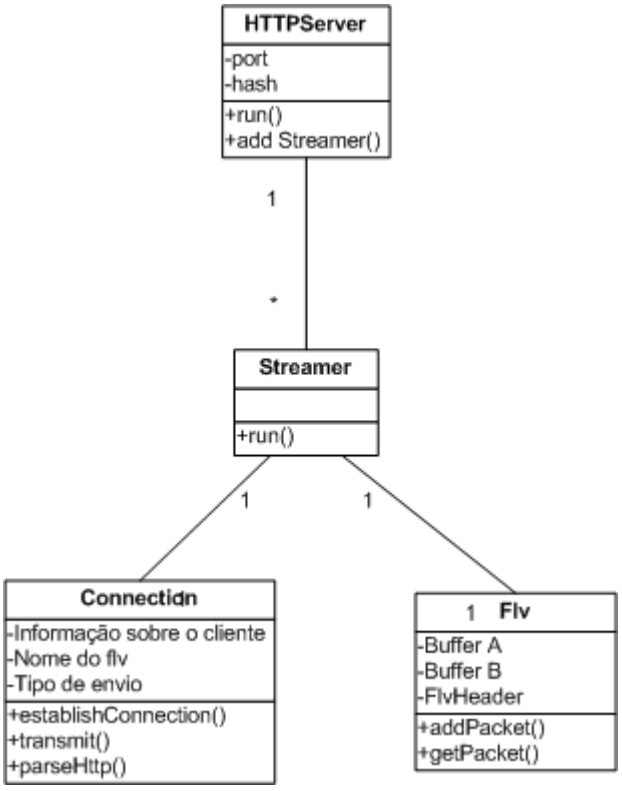

Figura 11 - Diagrama de Classes do Red5

A Figura 11 mostra o diagrama de classes com as classes principais do módulo construído para suportar a recepção de *streams*.

5.3.1 HTTPServer

Um objecto da classe HTTPServer (que estende a classe Thread do Java) é responsável pela recepção dos Post HTTP que virão da Aplicação Streamer. Guarda também uma *Hash Table* com os streamers activos, para que possa ser consultada quando houver uma requisição de um ficheiro FLV.

Esta classe encontra-se na camada "Servidor de *Streams*", do diagrama de arquitectura horizontal da Figura 6.

5.3.2 Streamer

Um objecto da classe Streamer (que estende a classe Thread do Java) é responsável por tudo o que diz respeito à recepção do vídeo da Aplicação Streamer. Tem um objecto Connection e um FLV. É também aqui que é feita a comunicação com as aplicações externas que fazem a transcodificação do vídeo para o formato FLV.

Esta classe, à semelhança da HTTPServer, encontra-se na camada "Servidor de *Streams*" do diagrama de Arquitectura Horizontal da Figura 6.

5.3.3 Connection

Um objecto da classe Connection guarda informação sobre a ligação estabelecida com a aplicação que envia o vídeo: informação sobre o cliente, o nome do FLV, o tipo de envio HTTP (chunked ou não).

A classe Connection encontra-se também na camada "Servidor de *Streams*" do diagrama de Arquitectura Horizontal, da Figura 6.

5.3.4 Flv

Como já dito, um objecto Streamer tem também um objecto FLV. Aqui fica guardada a informação relativa ao *stream FLV* que está a ser recebido. Esta classe tem dois buffers que contêm porções do vídeo FLV recebido. O primeiro buffer vai guardando os dados recebidos pelo objecto Streamer a que pertence, enquanto o segundo buffer guarda os dados que estão disponíveis para serem consultados pela Aplicação Receiver.

Temporariamente o conteúdo do primeiro buffer é copiado para o do segundo. Quanto menor for este intervalo de tempo, menos *delay* existirá entre o início da transmissão do vídeo e a recepção deste, mas fará com que o sistema seja menos tolerante a falhas.

Importa salientar que existe um número que vai sendo incrementado de cada vez que os buffers são copiados e que serve para que, quando um Receiver se conectar ao Servidor, possa saber que chunk (pedaço de um vídeo FLV que tem um tamanho igual ao buffer) está actualmente a ser transferido e ao compará-lo com o seu possa saber se necessita de requisitar esse chunk ou se já tem o mais actual.

A classe FLV encontra-se na camada "Gestor de Ficheiros" da Arquitectura Horizontal da Figura 6.

5.3.5 Sequência

A Figura 12 contém o diagrama de sequência dos principais módulos alterados do Servidor do Red5. Para a melhor visualização deste, as classes Streamer e Connection foram condensadas e representadas pela Streamer Class.

A comunicação entre o Servidor e o Streamer é estabelecida segundo o protocolo HTTP, sendo que é enviado por parte da Aplicação Streamer um Post indicando o seu desejo de enviar um vídeo. Caso o vídeo recebido esteja num formato que não o FLV, este é enviado para duas aplicações externas (VLC e FFmpeg) que fazem a seu transcodificação e o reenviam de novo para o Servidor. Para que o diagrama da Figura 12 fosse legível, foi omitida a representação do envio do vídeo da Classe Streamer para o Transcoder, sendo que este continua a existir.

Quando o vídeo chega finalmente no formato FLV é criado um objecto do tipo FLV (que guardará os dados que estão a chegar por *streaming*). Cada objecto FLV tem ainda um objecto da classe FLVReader que é responsável pela leitura dos dados do objecto FLV.

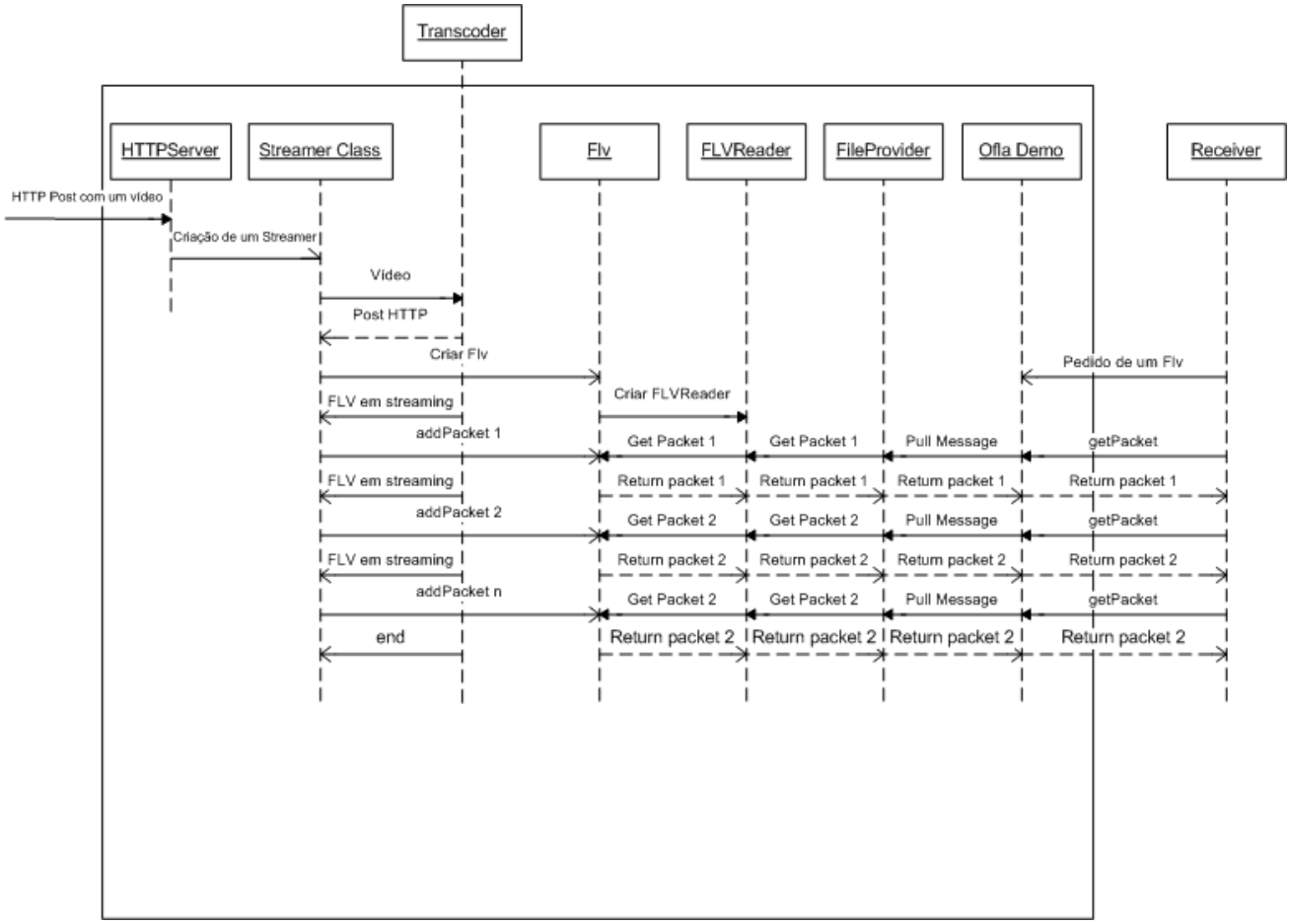

Figura 12 - Diagrama de Sequência do Red5

No momento em que um Receiver pede um ficheiro FLV, comunica com o Ofla Demo que irá confirmar a existência do ficheiro a uma hash com todos os streamers. Se o ficheiro existir, então o Receiver pedirá para que este lhe seja enviado por *streaming*. Importa salientar que a comunicação entre o Receiver e o Servidor é feita com a classe OflaDemo.

É importante notar que o Ofla Demo é uma aplicação construída para o Red5 e que tem dois componentes principais: uma aplicação em Java a correr sobre o Red5 e que consegue aceder às classes e bibliotecas destes, e cuja classe principal está representada na Figura 12; um ficheiro Swf que comunica com a outra componente do OflaDemo e que permite visualizar a listagem dos vídeos disponíveis no servidor, bem como os próprios vídeos, como se pode confirmar na Figura 13.

A classe responsável por lidar com o envio de cada *stream* para um Receiver é a FileProvider. Esta classe foi modificada para se adaptar às alterações do Red5 e que permitem que receba *streams*. Assim, cada objecto FileProvider guarda o número do chunk que o cliente tem, e faz constantemente Pooling ao FLVReader de forma a saber se existe algum chunk mais actual. Quando isso acontece, pede-o ao FLVReader, que por sua vez acederá ao FLV, para obter o chunk mais recente.

Este chunk será retornado até ao objecto FileProvider que tratará de preparar o seu envio por RTMP.

Existem duas situações de excepção a serem consideradas: a situação em que o streamer pára de enviar o vídeo (ou o vídeo termina ou houve algum problema com a ligação), e a situação em que o Receiver desliga.

A segunda situação já estava contemplada pelo Red5, pelo que não foi preciso alterar nada. Quando deixa de existir ligação com o Receiver o Servidor continua a funcionar com normalidade, deixando simplesmente de enviar dados para aquele Receiver, e fechando a ligação.

Figura 13 - Ofla Demo

Quando se deixa de receber dados do streamer, aconteceu uma de duas situações: ou o vídeo terminou ou houve algum problema com a ligação.

Caso tenha havido algum problema com a ligação e esta tenha falhado, o Servidor ficará à espera, durante um tempo definido, que volte a ser estabelecida. Caso tal aconteça voltará a enviar a disponibilizar o vídeo que receber.

Caso o vídeo tenha terminado ou haja uma quebra demorada na ligação, será finalizada a disponibilização do vídeo.

Realce-se o facto de um dos objectivos futuros do trabalho ser o reconhecimento automático dos conteúdos recebidos dos streamers. Este módulo de reconhecimento é independente e seria, caso fosse possível, fornecido pela ClusterMedia Labs. À data de conclusão do projecto não tinha havido disponibilidade de se construir este módulo, fazendo com que essa funcionalidade não tivesse sido integrada. Ainda assim, foi definida a arquitectura deste sistema e são apenas necessários alguns ajustes para que fique funcional.

O módulo de reconhecimento automático tinha como objectivo inicial ajudar a filtrar o conteúdo, indicando se existia ou não música no vídeo. Assim, para que tal possa ser integrado, a Classe Streamer está preparada para, ao enviar o ficheiro recebido do streamer para o VLC e para o FFmpeg para ser efectuada a transcodificação, enviar o vídeo já no formato FLV para um classificador que lhe inserirá *tags* indicando se existe, ou não, música no vídeo.

5.3.6 Visualização do Estado dos *Streams*

O caso de uso associado ao servidor (visualização do estado dos *streams*) foi implementado em todas as classes, recorrendo ao uso da biblioteca Log4J, da Apache. Esta biblioteca permite estruturar os tipos de Logging necessários para a aplicação, dividindo-os em níveis (Debug, Info, Warning, Error, Fatal) e utilizando ficheiros de configuração externos para um melhor controlo sobre quais as classes em que se quer efectuar o Logging.

Assim, foram inseridas várias mensagens a serem enviadas para o logging ao longo de todas as classes modificadas do Red5. Estas mensagens contêm, entre outras coisas, a informação que tinha sido especificada (número de *streams* em simultâneo, tempo há que estão a correr, indicação do IP do streamer, número de pacotes enviados e existência de falhas na ligação). Além disso, estas mensagens são ainda registadas num ficheiro de logging para consulta posterior.

5.4 Receiver

A Aplicação Receiver foi desenvolvida em Flash Lite 3.0 de forma a poder aproveitar a implementação do RTMP que o Flash Lite 3.0 traz.

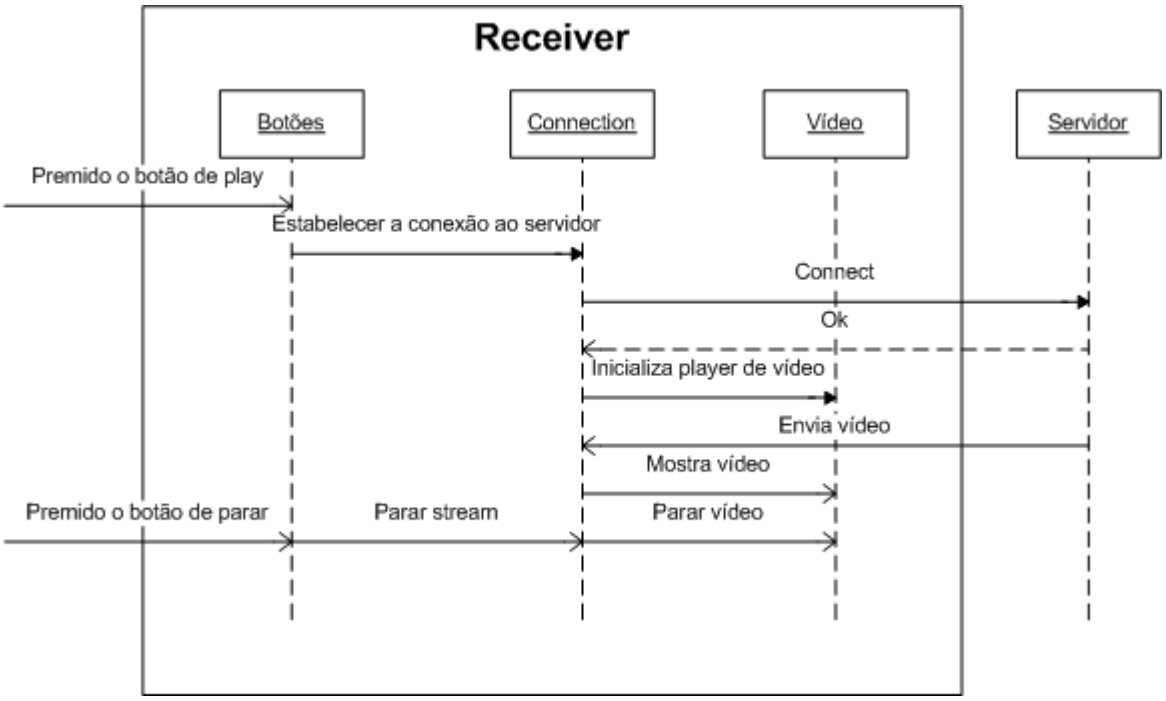

Figura 14 - Diagrama de Sequência do Receiver

A Figura 14 mostra o seu diagrama de sequência.

Quando é iniciada a aplicação é mostrado um menu com os botões de Play e Stop. Premindo o botão de Play é estabelecida a ligação ao Servidor e feito o pedido do vídeo (Figura 15, figura da esquerda). Após a ligação ter sido feita, é associado o vídeo remoto ao Player, estando assim pronto a ser recebido e visualizado. O comando netPlay acciona a recepção do vídeo e a sua visualização, sendo mostrado no fundo do ecrã o tempo decorrido desde o início da recepção do vídeo (Figura 15, figura da direita).

Se for necessário, é possível interromper a recepção e visualização do vídeo, premindo o botão de Stop.

Figura 15 - Emulador o Adobe Device Central a Correr a Aplicação Streamer

Capítulo 6

Testes e resultados

Cada um dos módulos foi testado da forma mais independente possível antes de serem realizados testes de integração, para que fosse possível identificar erros em cada um deles mais rapidamente.

Foram efectuados testes à medida que novas funcionalidades eram desenvolvidas, bem como vários testes depois de cada módulo estar concluído.

Apesar de não se ter seguido nenhuma metodologia formal de testes, estes foram cruciais durante o desenvolvimento. Os testes tiveram dois grandes objectivos: o primeiro foi o de detectar problemas para os poder corrigir; o segundo testar as tecnologias envolvidas e os dispositivos usados, para que se pudesse detectar as suas limitações e ponderar o seu uso numa aplicação comercial.

Houve uma maior ênfase nos testes relativos à Aplicação Streamer e ao Servidor, por diversos motivos:

- Foram os dois primeiros módulos a serem desenvolvidos, havendo mais tempo para se efectuarem os testes;
- O módulo do Servidor foi também usado noutros projectos na ClusterMedia o que fez com que fosse importante garantir desde logo o seu correcto funcionamento. Levou também a que houvesse um maior número de testes, já que outros funcionários da empresa foram utilizando este módulo e ajudando a encontrar falhas;
- A Aplicação Receiver é mais simples do que as restantes, existindo aplicações semelhantes na ClusterMedia Labs que já tinham sido intensamente testadas.

Serão agora listados os testes efectuados a cada um dos módulos separadamente.

6.1 Aplicação Streamer

6.1.1 Testes

Houve duas fases de testes relativos à Aplicação Streamer, a primeira efectuada recorrendo-se ao emulador do JWTK, e segunda no Nokia N80.

Assim, na primeira fase foi inicialmente testada a funcionalidade de envio de um ficheiro que se encontrava no dispositivo móvel (neste caso, um emulador) para um Servidor a correr localmente. Não foi detectado nenhum problema nos vários testes, sendo o ficheiro entregue correctamente e sem falhas.

Posteriormente, os testes passaram a ser realizados no Nokia N80. Esta mudança foi motivada pela limitação do emulador do JWTK que não permite acesso à câmara. Foram então feitas e testadas pequenas aplicações que capturavam fotografias e vídeo e as guardavam no sistema de ficheiros do Nokia N80.

Os testes a serem efectuados posteriormente concentraram-se no envio e correcta recepção dos vídeos para o Servidor, desta vez – e ao contrário do que acontecia quando a aplicação corria no emulador – a comunicação com o Servidor era feita via wireless. As falhas de comunicação foram uma constante durante estes testes, como explicado na secção Resultados.

Por fim foram efectuados testes à aplicação finalizada. Estes testes consistiram na execução da aplicação várias vezes para se tentarem encontrar bugs mas também para testar a fiabilidade do sistema quando em situações menos ideais da ligação wireless.

Como nota, importa sublinhar que o processo de testes é bastante moroso. É necessário compilar o projecto em Java ME para um jar, fazer a instalação deste no telemóvel, conceder-lhe permissões e posteriormente colocar o programa a correr. Estas operações gastam quase 10 minutos, o que dificultou bastante a fluidez dos testes. Por esta razão foi dada preferência à realização dos testes no emulador, em detrimento dos testes no Nokia N80, sempre que assim era possível.

6.1.2 Resultados

Tempos de envio:

No emulador, o tempo necessário para o início de envio dos dados nunca excedeu os três segundos, o que foi considerado um tempo aceitável. No entanto estes tempos foram apenas indicadores, já que o que de facto interessa são os tempos obtidos num dispositivo real.

Existem dois intervalos de tempo importantes a serem tidos em consideração para se avaliarem os resultados:

- O primeiro é entre o momento em que a câmara termina a gravação e os dados são recebidos pela Thread de *streaming* até ao momento em que é perguntado ao utilizador se autoriza a conexão à Internet;

- O segundo é desde que a ligação é autorizada pelo utilizador até à altura em que o pedido é recebido no Servidor.

O Nokia N80 demorou sempre mais de 5 segundos no primeiro momento e mais de 7 no segundo momento. O primeiro valor variou pouco em todos os testes realizados, sendo que o segundo já estava mais ligado ao tipo de rede wireless a que se conecta e à distância até ao Access Point.

Quando a ligação existente ao AP (Access Point) era forte não se verificaram perdas de frames nem quebras na ligação, não tendo acontecido o mesmo quando a ligação era fraca, existindo demoras no envio dos dados e quebras na ligação ao Servidor (houve mesmo situações em que a ligação à rede wireless foi também perdida).

Estes resultados foram interpretados como estando principalmente relacionados com as capacidades do Nokia N80. As constantes quebras de ligação existem também quando se utiliza este telemóvel para acesso à Internet, levando assim a crer que este problema transcende a aplicação (não houve a hipótese de testar esta teoria visto não se ter tido acesso a outros dispositivos móveis). Com a evolução dos telemóveis existe a possibilidade desta situação ser melhorada no futuro.

Qualidade do vídeo:

O facto de ter sido necessário recorrer ao "ligar/desligar" da câmara para proceder ao envio do que já fora filmado fez com que fossem perdidos vários segundos de filme (cerca de três segundos de cada vez que esta operação era feita com a câmara). No N80, o tempo máximo registado em que foi possível filmar ininterruptamente sem que houvesse falhas de memória foi de 15 segundos. O que dá um total de 3 segundos perdidos em cada 18, não sendo assim gravados cerca de 17% dos frames.

Esta limitação torna o lançamento de uma aplicação comercial baseada nesta plataforma inviável.

Os resultados relativos à qualidade do vídeo não são também os melhores. A Figura 16 mostra nove frames de um vídeo obtido pela câmara do Nokia N80 e transmitidos para o Servidor. O vídeo tem resolução de 176x144, 15 frames por segundo, e utiliza o *codec* h263 e encontra-se no formato 3gp. Todas estas características são mantidas quando o vídeo é convertido para o formato FLV, não se notando perdas na qualidade. O tamanho do vídeo, por seu lado, é substancialmente diferente entre o formato 3gp e o FLV, sendo que o primeiro ocupa 115KB e o segundo 602KB.

Figura 16 - Frames de um Vídeo Capturado pela Aplicação Streamer

Como se observa, a qualidade e a resolução do vídeo não são os melhores, sendo esperado que num futuro próximo os telemóveis tragam câmaras com melhor resolução. Espera-se também que as ligações wireless se tornem melhores (caso contrário, mesmo que haja melhoria nas câmaras, não existirá a possibilidade de enviá-los, já que os vídeos ocuparão mais espaço e consequentemente mais largura de banda).

Durante os testes descobriu-se também que os POST HTTP enviados pelo emulador e pelo Nokia N80 variavam. Enquanto o que era enviado pelo emulador enviava os dados utilizando chunks (propriedade do protocolo HTTP 1.1 que permite que sejam enviados os dados de forma fragmentada e que é particularmente útil quando não se sabe o tamanho dos dados a enviar) o Nokia N80 não o fazia.

6.2 Servidor

6.2.1 Testes

Foram efectuados diversos testes ao Servidor de Red5 ao longo do período que decorreu o projecto. Visto já existir anteriormente uma versão modificada deste Servidor, esta foi executada diversas vezes com o objectivo de se lhe descobrirem problemas. No entanto, esta

versão não tinha um mecanismo de debug que permitisse uma análise posterior, estando o debug a ser feito recorrendo à escrita de informação para a consola.

Assim um dos primeiros passos foi a integração de uma biblioteca de debug da Apache, o log4j, que permite um maior controlo sobre o nível de debug efectuado assim como o registo para ficheiros.

Estes ficheiros ajudaram a acelerar o desenvolvimento, já que se tornou mais fácil identificar os problemas que iam surgindo.

Posteriormente foi integrado o Servidor HTTP e feitos diversos testes de stress à aplicação, que incluíram a recepção de vários *streams* durante alguns dias, o envio de informação não suportada pelo protocolo ou pelo transcoder (cabeçalhos HTTP errados, vídeos com erros) e falhas na ligação.

De notar que o tempo de teste foi sempre uma adversidade já que para compilar e lançar o Red5 eram necessários quase dois minutos.

6.2.2 Resultados

Os resultados obtidos relativamente ao Servidor foram bastante satisfatórios. Em todos os testes de stress efectuados não houve nenhum problema a assinalar tendo o sistema funcionado como esperado.

Num primeiro teste, o Servidor foi deixado a correr durante dois dias estando a receber três *streams* de entrada contínuos (provenientes de três canais de televisão online). Não foram registadas nos logs de debug quaisquer falhas, tendo todas as transmissões funcionado sem problemas. Este teste foi realizado num QuadCore e teve um máximo de 5 clientes a requisitarem os vídeos disponibilizados.

Os testes realizados em que se interrompeu o envio dos *streams* para o Servidor foram também bem sucedidos, já que assim que foi retomado o envio os vídeos passaram a estar novamente disponíveis no Servidor.

Os testes em que foram enviados pedidos sem significado para o Servidor mostraram que este se encontrava preparado para lidar com eles, ignorando-os.

Quanto à capacidade de processamento necessária para correr o Servidor foi também possível tirar algumas conclusões. A maioria dos testes realizados (excepção feita ao teste em que se deixou o Servidor a correr durante dois dias), foram efectuados num computador portátil ASUS F3JC com um Intel Core2Duo 1.6 MHz, e um giga de RAM, a correr o Windows XP SP2.

Foi testado o desempenho desse computador ao correr o Servidor em diversas condições. Na Tabela 5 encontram-se os resultados obtidos relativamente à utilização do processador e da memória em diferentes estados do funcionamento do Servidor.

A realização destes testes seguiu a seguinte ordem:

1. Foi iniciado o computador;
2. Foi lançado a nova versão do Servidor Red5;
3. O Servidor terminou a inicialização.

4. Foi adicionado o primeiro Streamer (e feito a transcodificação dos dados por ele enviados, utilizando para isso o VLC e o FFmpeg).
5. Foi adicionado o primeiro Receiver (que se conectou a partir de outra máquina);
6. Foi adicionado o segundo Receiver;
7. Foi adicionado o terceiro Receiver;
8. Foi adicionado o quarto Receiver;
9. Foi adicionado o segundo Streamer;
10. Foi adicionado o terceiro Streamer;
11. Foi adicionado o quarto Streamer.

Ao analisar-se a Tabela 5 verifica-se que o funcionamento da versão do Red5 que foi desenvolvida durante o projecto utiliza cerca de 70 MB de RAM e aumenta a necessidade de processamento do F3JC em cerca de 4,5%.

Os resultados restantes são mais facilmente perceptíveis analisando a Figura 17 e 18.

Tabela 5 - Desempenho do F3JC a correr o Servidor

Exp	Red5	Streamers	Receivers	Memória (MB)	Proc min (%)	Proc max (%)	Proc med (%)
1	Não iniciado	-	-	436	1	2	1,5
2	A iniciar	-	-	467	30	70	50
3	Iniciado	0	0	506	2	10	6
4	Iniciado	1	0	559	15	25	20
5	Iniciado	1	1	564	15	25	20
6	Iniciado	1	2	568	15	30	22,5
7	Iniciado	1	3	574	15	30	22,5
8	Iniciado	1	4	580	15	30	22,5
9	Iniciado	2	4	628	25	50	37,5
10	Iniciado	3	4	676	30	70	50
11	Iniciado	4	4	722	50	75	62,5

A Figura 17 mostra a relação entre o processamento médio e o número de Streamers e Receivers. Pode verificar-se que, à medida que o número de Streamers cresce, a necessidade de processamento cresce também – de uma forma quase constante – e ronda os 12,5%.

Importa notar que a transcodificação dos dados recebidos dos Streamers está a ser feito na mesma máquina onde se encontra a correr o Red5, sendo que o processamento extra que existe quando é associado um novo Streamer se deve a este facto.

Por outro lado, a adição de Receivers parece não ter qualquer efeito sobre a necessidade de processamento. A única variação significativa visível na Figura 17 (existente entre a adição

do Receiver número 2 e do Receiver número 3) deve-se ao facto de se terem adicionado vários Streamers entretanto.

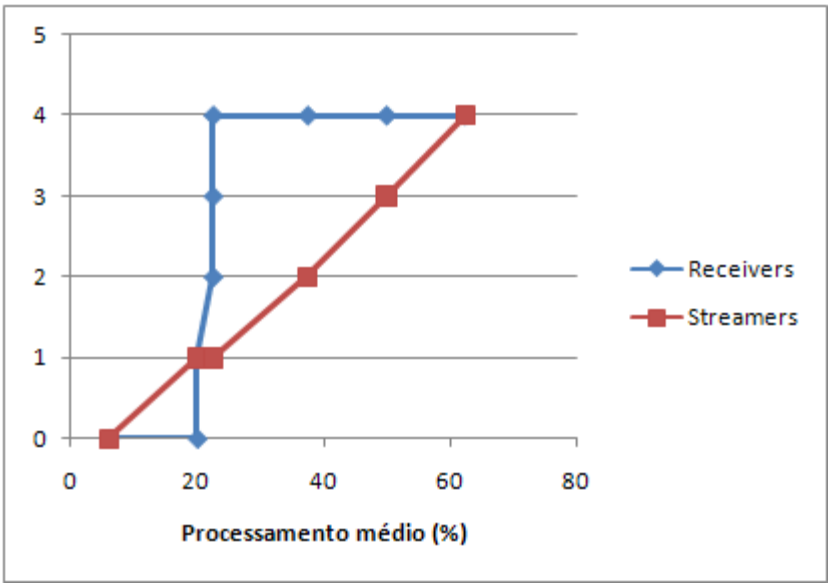

Figura 17 - Relação entre o Número de Streamers e Receivers e a Processamento do Asus F3JC

A Figura 18 mostra a relação existente entre o número de Streamers e Receivers, e a memória RAM utilizada.

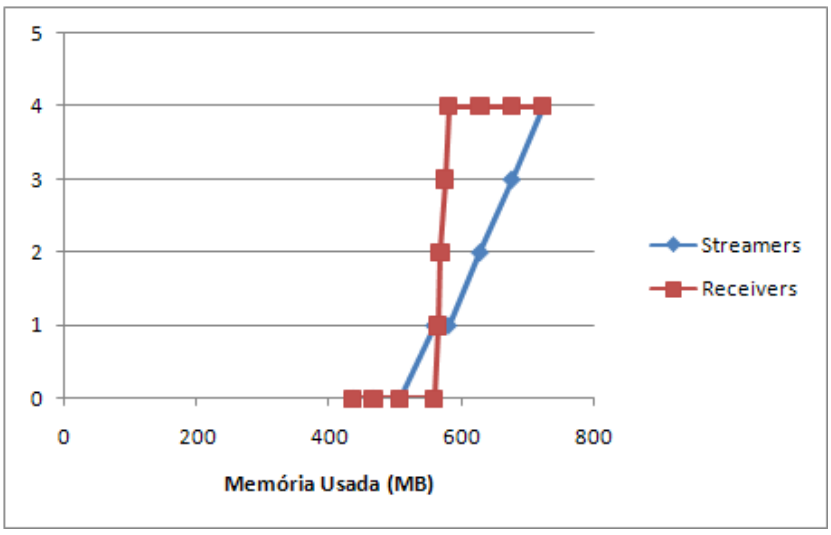

Figura 18 - Relação entre o Número de Streamers e Receivers e a Memória Utilizada

Novamente, a adição de Receivers não parece ter um efeito grande na quantidade de RAM necessária para o funcionamento do Servidor. Analisando a Tabela 18 constata-se que são utilizados em média cerca de 4MB de RAM extra por cada Streamer conectado.

Relativamente às necessidades de memória associados a cada streamer, existe um aumento de cerca de 48MB por cada um que é adicionado. Estes valores prendem-se sobretudo pelas necessidades de memória que as aplicações VLC e FFmpeg têm.

6.3 Receiver

6.3.1 Testes

A Aplicação Receiver foi sujeita a um menor número de testes e, devido ao seu tamanho reduzido, foi apenas testados quando se encontrava já completa.

Inicialmente foi apenas testada a aplicação a correr no emulador, estando este a correr na mesma máquina que o Servidor. Os testes foram todos bem sucedidos, não se tendo detectado nenhum problema.

Posteriormente foi feito a instalação da aplicação para o Nokia N80, não existindo também problemas a assinalar.

6.3.2 Resultados

Os resultados obtidos no Nokia N80 revelaram-se satisfatórios, tendo os vídeos sido recebidos sem que houvesse qualquer problema. O tempo de conexão da aplicação ao Servidor foi também satisfatório, sendo um pouco menor que o utilizado pela aplicação Streamer.

6.4 Integração

6.4.1 Testes

Visto não haver dois dispositivos móveis, não foi possível testar todo o sistema com este a correr no *hardware* para o qual foi desenhado. No entanto, foi possível testá-lo com recorrendo a emuladores (quer para o streamer quer para o Receiver).

Foram assim efectuados dois testes: o primeiro com a aplicação Streamer a correr no emulador do JWTK, e a enviar um vídeo que se encontrava no seu sistema de ficheiros; o segundo teste com a aplicação Receiver a correr no emulador do Adobe Device Central, estando a Aplicação Streamer a correr no Nokia N80.

6.4.2 Resultados

Não houve nenhuma situação inesperada, já que todos os módulos do sistema tinham sido previamente testados juntamente com os módulos com os quais comunicam.

Ambos os testes produziram os resultados esperados, sendo que o tempo de *delay* no primeiro teste foi um pouco inferir ao do segundo teste, já que o tempo gasto para a ligação da

Aplicação Streamer ao Servidor, estando esta a correr no emulador, é menor que o tempo gasto para o Nokia N80 efectuar a mesma operação.

Capítulo 7

Conclusões e Trabalho Futuro

Este capítulo tem como objectivo fazer uma análise final do projecto, analisando-se os requisitos iniciais e o grau de cumprimentos destes, fazendo-se uma análise aos resultados obtidos, e perspectivando o trabalho que poderia ser ainda desenvolvido.

Far-se-á também uma breve descrição e análise da adaptação do ambiente académico para o empresarial.

7.1 Cumprimento de Requisitos

Todos os requisitos funcionais propostos inicialmente foram cumpridos, inclusivamente os cuja prioridade era média ou baixa, como pode ser observado na Tabela 6.

Tabela 6- Requisitos Funcionais

Módulo	**Descrição da funcionalidade**	**Prioridade**	**Estado**
Streamer	Capturar o vídeo de um dispositivo móvel em tempo real e enviá-lo por *streaming* para um Servidor em tempo real.	Alta	Implementado
Streamer	Suspender a paragem e o envio do vídeo.	Média	Implementado
Streamer	Escolher o IP do Servidor.	Média	Implementado
Streamer	Guardar o vídeo localmente.	Baixa	Implementado
Servidor	Visualizar o estado dos Input *Streams*	Alta	Implementado
Receiver	Receber e visualizar um vídeo no formato FLV.	Alta	Implementado
Receiver	Parar o vídeo.	Média	Implementado
Receiver	Visualizar o tempo de vídeo	Baixo	Implementado

Também os requisitos não funcionais foram cumpridos:

- Foi construído o sistema pretendido, com os três módulos a funcionarem em plataformas independentes e completamente integrados;
- Foram utilizadas as tecnologias e plataformas requeridos;
- Foram utilizados os protocolos e normas de comunicação requeridos;
- O Servidor foi modificado para poder receber vários *streams* de entrada;
- O Servidor encontra-se robusto e fiável;
- Tempos de *delay* foram abaixo dos requeridos.

Assim sendo, o sistema desenvolvido assemelha-se ao pretendido, não havendo nenhum problema a assinalar.

7.2 Análise de Resultados

Os resultados relativos aos testes foram analisados no Capítulo 6. Este subcapítulo pretende tirar conclusões mais abrangentes sobre o projecto.

7.2.1 Tecnologias

O estudo das várias plataformas para o desenvolvimento da Aplicação Streamer permitiu que as decisões tomadas relativas às tecnologias a usar no futuro fossem mais bem fundamentadas e suportadas. Para um sistema como o LiveXtend será fundamental que as tecnologias para a Aplicação Streamer tenham um bom acesso aos recursos do sistema, tendo esta sido a principal limitação do Java ME. No futuro, dever-se-á ponderar a hipótese da utilização do Python, ou mesmo do Open C++ para Symbian, mesmo que isso implique maiores gastos no desenvolvimento da aplicação.

Possivelmente o lançamento de aparelhos com o Google Android nativo abrirá novas portas ao desenvolvimento de aplicações para dispositivos móveis, sendo esta uma possível futura boa opção.

As restantes tecnologias utilizadas adequaram-se ao projecto. O uso do Red5 representa uma mais-valia em relação ao Adobe Media Server já que, além de ser gratuito, permite a modificação do seu código e a criação rápida de aplicações que usufruam das potencialidades do Red5.

7.2.2 Metodologia de Testes

Uma das maiores dificuldades encontradas prendeu-se com o tempo gasto para se efectuarem testes. Nas plataformas móveis, o tempo para a instalação das aplicações no dispositivo móvel (e mesmo o tempo necessário para o lançamento e arranque do emulador)

eram um bottleneck no desenvolvimento. A somar a isto têm-se as limitações de processamento do Nokia N80.

Foi complicado determinar uma metodologia de testes eficaz para esta situação. Este caso deverá ser analisado no futuro para diminuir os tempos gastos em desenvolvimento e teste.

Além do dispositivo móvel, também a compilação e lançamento do Red5 é um processo demorado, e podia demorar até três minutos.

7.2.3 Mais-valias

A realização do trabalho trouxe mais-valias indiscutíveis quer para o autor do projecto, quer para a empresa onde foi desenvolvido.

A versão do Red5 criada durante o projecto é já a versão padrão usada na ClusterMedia Labs e foi também anunciado um possível desenvolvimento de uma solução com fins mais comerciais do sistema LiveXtend.

Para o autor do projecto, este revelou-se bastante enriquecedor, quer ao nível de aquisição de conhecimentos tecnológicos, como o do mercado das tecnologias móveis.

O contacto com a realidade empresarial ajudou a alargar horizontes e ter um maior conhecimento da forma como as empresas se organizam e do papel a desempenhar nelas.

7.3 Trabalho Futuro

Apesar de terem sido implementados todos os requisitos funcionais previstos no início do projecto, existem bastantes outras funcionalidades que poderão ser implementadas no futuro e que não foram desenvolvidas para o projecto principalmente devido ao limite de tempo para a sua realização.

De seguida encontram-se listadas as funcionalidades que poderão ser desenvolvidas no futuro e que trarão mais-valias para o sistema:

- A integração do LiveXtend com um sistema de informação que fizesse:
  - Gestão de utilizadores, com área pessoal;
  - Disponibilização posterior dos vídeos;
  - Envio de alertas para utilizadores;
  - Pesquisa de vídeos;
  - Catalogação dos vídeos.
- A integração efectiva com um sistema de análise de vídeo e áudio;
- A integração do LiveXtend com as diversas Social Networks existentes (Hi5, FaceBook, MySpace, Orkut) ou a criação de uma Social Network assente no LiveXtend.
- Geolocalização dos vídeos;

- Utilização de outras normas de comunicação (GSM, 3G);
- Utilização de outras linguagens de programação para o desenvolvimento do Streamer (Python ou uma linguagem com maior acesso às funcionalidades do dispositivos móveis)
- Criação de um módulo Streamer para Desktops e Portáteis que use as câmaras destes.

Bibliografia

[Ado07] Adobe. , 2007, . Adobe. Available at http://www.adobe.com/aboutadobe/pressroom/pressreleases/200709/100107FlashLite3.html, last accessed on Jun. 2008

[All08] All App Labs. All App Labs. Available at http://www.allapplabs.com/log4j/log4j_levels.htm, last accessed on Jul. 2008

[Ant05] A. Pranata. , 2005, . Antony's Mobile Blog. Available at http://mobile.antonypranata.com/?p=12, last accessed on 2008

[ASa06] A. San Juan. , 2006. The Lurker's guide to Java ME CDC. Available at http://www.blueboard.com/javame/intros.htm, last accessed on Jun. 2008

[Câm08] Câmara Municipal de Amarante. Amarante, Cidade Wireless. Available at http://www.cm-amarante.pt/document/808306/885436.pdf, last accessed on Jun. 2008

[Câm081] Câmara Municipal de São João da Madeira. São João da Madeira, Capital Wireless. Available at http://www.cm-sjm.pt/index.php?oid=7770&op=all, last accessed on Jun. 2008

[Can08] Canalys. Canalys. Available at www.canalys.com/pr/2008/r2008021.htm, last accessed on Jul. 21, 2008

[CEV08] CEVA. CEVA. Available at http://ceva-dsp.mediaroom.com/index.php?s=glossary, last accessed on Jun. 2008

[Dav05] D. Needle. , 2005, . Wi-Fi Planet. Available at http://www.wi-fiplanet.com/news/article.php/3551686, last accessed on Jun. 2008

[Ecl05] Eclipse ME. , 2005. Eclipse ME. Available at http://eclipseme.org/, last accessed on Jun. 2008

[Fei08] F. Mosleh. , 2008, . Video/Imaging Design line. Available at http://www.videsignline.com/howto/208400992, last accessed on Jun. 2008

[Fla07] Flash Lite. , 2007, Mar.. Adobe. Available at http://www.adobe.com/products/flashlite/version/, last accessed on 2008

[Goo08] Google. Android Emulator. Available at http://code.google.com/android/reference/emulator.html#limitations, last accessed on 2008

[Goy08] V. Goyal. java.net The Source for Java Technology Collaboration. Available at http://today.java.net/pub/a/today/2005/02/09/j2me1.html, last accessed on Jun. 2008

[Hri051] Hrissan. , 2005, Jan.. Symbian OS Design Faults. Available at http://www.codeproject.com/KB/mobile/Symbian_OS_design_faults.aspx?print=true, last accessed on Jun. 2008

[Jav08] Java Community Process. Java Community Process. Available at

http://jcp.org/en/jsr/overview, last accessed on Mar. 2008

[JoB06] J. Best. , 2006, . Networks - Breaking Business and Technology News at silicon.com. Available at http://networks.silicon.com/mobile/0,39024665,39156391,00.htm, last accessed on Jun. 2008

[Jua06] A. S. Juan. , 2006. The Lurker's Guide to Java ME CDC. Available at http://www.blueboard.com/javame/intros.htm, last accessed on Jun. 2008

[Kun07] KuneriLite. , 2007. Kuneri Lite. Available at http://www.kunerilite.net/, last accessed on 2008

[Lar08] L. Rudolph. Symbian development general notes. Available at http://org.csail.mit.edu/mode/index.php/Symbian_development_general_notes#Why_not_use_C.2B.2B, last accessed on 2008

[Mob07] Mobile Gazette. , 2007, . Mobile Gazette. Available at http://www.mobilegazette.com/blackberry-pearl-8120-curve-8320-07x10x19.htm, last accessed on Jul. 2008

[Nok08] Nokia. , 2008, Mar.. Forum Nokia, Developer Community Wiki. Available at http://wiki.forum.nokia.com/index.php/Java_ME, last accessed on Jun. 2008

[Ope08] Open Source Flash. Open Source Flash. Available at http://osflash.org/red5, last accessed on 2008

[Reu07] Reuters. , 2007, Nov.. Reuters. Available at http://investing.reuters.co.uk/news/articleinvesting.aspx?type=media&storyID=nL29172095, last accessed on Jun. 2008

[Rob08] S. Robinson. , 2008, Jan.. Strategy Analytics. Available at http://www.strategyanalytics.com/default.aspx?mod=ReportAbstractViewer&a0=3727, last accessed on 2008

[Roc08] D. Rocha. , 2008, . Daniel Rocha's Forum Nokia Blog. Available at http://blogs.forum.nokia.com/blog/daniel-rochas-forum-nokia-blog/web-run-time-wrt/2008/05/30/which_technology_c, last accessed on Jun. 2008

[Roc081] D. Rocha. , 2008, . Daniel Rocha's forum Nokia blog. Available at http://blogs.forum.nokia.com/blog/daniel-rochas-forum-nokia-blog/web-run-time-wrt/2008/05/27/which_technology, last accessed on Jun. 2008

[Roc88] D. Rocha. , 2008, . Which technology should I use for development? Round 1: Java. Available at http://blogs.forum.nokia.com/blog/daniel-rochas-forum-nokia-blog/web-run-time-wrt/2008/05/27/which_technology, last accessed on Jun. 2008

[Sco02] S. Storkel. , 2002, Nov.. ONJava. Available at http://www.onjava.com/pub/a/onjava/2002/12/11/eclipse.html, last accessed on Jun. 2008

[Sri07] S. Roy. , 2007, . Java World. Available at http://www.javaworld.com/javaworld/jw-09-2007/jw-09-mobilevideo2.html?page=1, last accessed on Jul. 2008

[Sun08] Sun Developer Network. Sun Developer Network. Available at http://java.sun.com/products/mmapi/, last accessed on Mar. 2008

[Sun081] Sun Developer Network. Sun Developer Network. Available at http://java.sun.com/products/sjwtoolkit/, last accessed on 2008

[Sym07] Symbian Press. , 2007. Essential S60 Developers' Guide. Available at http://www.forum.nokia.com/info/sw.nokia.com/id/80dc01fa-2260-49ca-8ee3-f0a414adb78a/Essential_S60_Developers_Guide.pdf.html

[The05] , 2005, . Mobile analysis and development. Available at http://bonte.co.uk/myBlog/?p=145, last accessed on 2008

[The08] The Apache Ant Project. , 2008, Jun.. The Apache Ant Project. Available at

http://ant.apache.org/faq.html#what-is-ant, last accessed on Jul. 2008

[Tom08] T. Krazit. , 2008, Mar.. Cnet News. Available at http://news.cnet.com/8301-13579_3-9906697-37.html, last accessed on Jun. 2008

[Wik084] Wikipedia. Wikipedia. Available at http://en.wikipedia.org/wiki/MIDlet, last accessed on Jun. 2008