Faculdade de Engenharia da Universidade do Porto

Mobile Streaming – Qualidade de Experiência

José Fernando Pereira dos Santos

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações, Especialização Comunicações Multimédia

Orientador: Prof. Dr. José Ruela Fernandes
Orientador: Eng. Pedro Nuno Pinto
Co-orientador: Prof.ª Dr.ª Maria Teresa Andrade

21 de Julho de 2010

© José Santos, 2010

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES **2009/2010**

A Dissertação intitulada

"MOBILE STREAMING - QUALITY OF EXPERIENCE"

foi aprovada em provas realizadas em 21/Julho/2010

o júri

Presidente Professor Doutor Luís António Pereira de Meneses Corte-Real
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Paula Viana
Professora Doutora Paula Maria Marques Moura Gomes Viana
Professora Coordenadora S/Agregação do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia do Porto

Professora Doutora Maria Teresa Magalhães da Silva Pinto de Andrade
Professora Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto (Orientador).

José António Ruela Simões Fernandes
Professor Doutor José António Ruela Simões Fernandes
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto (Co-Orientador).

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

José Santos
Autor **José Fernando Pereira Santos**

Faculdade de Engenharia da Universidade do Porto

Resumo

À medida que os pedidos por conteúdos multimédia através de computadores, televisão ou terminais móveis aumentam drasticamente, também os níveis de exigência por uma qualidade de experiência satisfatória por parte do utilizador se elevam. Infelizmente, a largura de banda limitada, o mercado de dispositivos e reprodutores multimédia muito fragmentado e em constante actualização, e ainda a imprevisibilidade do tráfego e condições de rede dificultam aos operadores o envio de conteúdos multimédia com qualidade correspondente às expectativas dos utilizadores. A Portugal Telecom Inovação dispõe de uma nova solução para *Mobile TV*, designada por WSS (Windless Streaming Server) que assegura, para além da funcionalidade básica de transmissão de vídeo para terminais móveis, a capacidade de realizar manipulação de *streams* de áudio e vídeo. Esta capacidade inclui a alteração do *bit rate* dos dados enviados para um terminal em tempo real. Com esta dissertação pretende-se recorrer às funcionalidades de manipulação de *streams* para melhorar a qualidade de experiência (QoE) do utilizador final, adaptando em tempo real a transmissão de áudio e vídeo às reais condições da rede. Inicialmente são descritos os principais conceitos relacionados com o *streaming* de conteúdos multimédia. De seguida, são identificados serviços e soluções existentes, sendo realizada uma comparação entre os mesmos no âmbito desta dissertação. Posteriormente, são expostas as tecnologias e os componentes utilizados na implementação da solução e justificadas as decisões tomadas no decorrer do seu desenvolvimento. É feita uma descrição da arquitectura da solução, e ainda a especificação de todos os seus componentes. Seguidamente é apresentada a prova de conceito realizada através de uma votação feita por um grupo de participantes externos, consistindo na classificação dos resultados visuais obtidos através da aplicação da solução desenvolvida. Finalmente são retiradas as conclusões à solução proposta.

Abstract

As demand for accessing multimedia content on computers, television, and mobile phones increases dramatically, so does user insistence on a satisfying viewing experience. Unfortunately, limited bandwidth, a highly fragmented device and media players market, and transient, unpredictable traffic and network conditions currently limit operators' ability both to deliver the media and meet user expectations for quality. Portugal Telecom Inovação has developed a solution for Mobile TV, designated WSS (Windless Streaming Server) which assures, besides the basic video transmission for mobile terminals, the ability to manipulate audio and video streams. This functionality includes changing the streams bit rate on the fly. This essay seeks to improve the Quality of Experience (QoE) for end-users relying on stream manipulation features, provided by the WSS to adapt the streaming audio and video to the conditions of a network connection. Initially, the main concepts associated with multimedia streaming are described. Then, current services and solutions are identified and compared in the context of this essay. The technologies and components used in the development of the solution are also presented, as well as the decisions made during the conception process followed by the architecture and specification of all the components of the developed module. Later, the proof of concept which consisted in the evaluation of the solutions' output by a group of external participants is presented together with the final conclusions to the presented solution.

Agradecimentos

No decorrer deste trabalho foram muitas as pessoas que apoiaram, incentivaram e colaboraram na sua realização, às quais não posso deixar de manifestar o meu mais sincero agradecimento. Um primeiro agradecimento dirigido aos orientadores desta dissertação de mestrado:

- ao Professor Doutor José Ruela, por toda a ajuda prestada, sempre disponível para todos os esclarecimentos não só no decorrer desta dissertação, mas também durante o meu percurso acamdémico;
- à Professora Doutora Maria Teresa Andrade pelo interesse demonstrado e motivação transmitida em todos os trabalhos realizados sob a sua supervisão;
- ao Engenheiro Pedro Pinto pelos vastos conhecimentos transmitidos e pelo apoio prestado dia após dia, esclarecendo todas as dúvidas que naturalmente surgiram.

Aos meus colegas e amigos da PT Inovação que todos os dias me proporcionaram um ambiente de trabalho irrepreensível, em especial ao Eng. Joaquim Azevedo, Eng. Nuno Ferreira, Eng. Alexandre Sapage, Eng. Luís Almeida, Eng. Luís Reis e Paulo Moura. Por último, um agradecimento ao Eng. Miguel Barros pela constante disponibilidade e paciência, ajudando-me sempre que necessário a superar alguns contratempos.

Aos meus colegas de faculdade por todos os bons momentos passados, e pela constante cooperação nas mais diversas àreas. Deixo ainda uma palavra de apreço à Faculdade de Engenharia da Universidade do Porto por ser uma instituição que, sem sombra de dúvida, mudou a minha vida tanto pelo que aprendi como pelos amigos que fiz.

Finalmente, quero agradecer à minha namorada Joana pelo constante apoio ao longo destes anos, e à minha família, em especial aos meus pais e irmão por toda a confiança e segurança que sempre me transmitiram em todos os momentos da minha vida, bons ou maus, estiveram sempre ao meu lado a proporcionar-me momentos de grande felicidade. Sem eles esta dissertação não seria possível.

"We cannot change the cards we are dealt,
just how we play the hand"

Randy Pausch, last lecture

Índice

Resumo v

Abstract vii

Agradecimentos ix

Índice xiii

Lista de figuras xv

Lista de tabelas xvii

Abreviaturas xix

Capítulo 1 1

Introdução 1
1.1 - Conteúdos multimédia 1
1.2 - Streaming 2
1.3 - QoE (Quality of Experience) 3
1.3.1 - QoS (Quality of Service) 4
1.4 - Motivação 5
1.5 - Objectivos 5
1.6 - Organização do documento 6

Capítulo 2 9

Estado da Arte 9
2.1 - Multimedia Streaming 9
2.1.1 - Diferentes Abordagens 9
2.1.2 - RTSP (Real-Time Streaming Protocol) 13
2.1.3 - HTTP-Live Streaming 16
2.1.4 - Smooth Streaming 19
2.1.5 - Adobe Dynamic Streaming 22
2.1.6 - Análise Comparativa 27
2.2 - Necessidade da Solução 29

Capítulo 3 31

Tecnologias e Componentes 31
3.1 - WSS (Windless Streaming Server) 31
3.2 - Protocolo de streaming utilizado 32
3.3 - Protocolo de transporte - TCP vs UDP 33

3.3.1 - TCP (Transmission Control Protocol) 33
3.3.2 - UDP (User Datagram Protocol) 34
3.3.3 - Decisão 35
3.4 - Aplicação no cliente 36
3.5 - Estimação da largura de banda utilizada 36
3.6 - *Interarrival Jitter* 37
3.7 - Linguagem Lua 39

Capítulo 4 41

Arquitectura e Implementação 41
4.1 - Concepção 41
4.1.1 - Entidades do WSS 42
4.1.2 - Entidades do módulo 43
4.1.3 - Configuração 43
4.1.4 - Diagrama de sequência do módulo 44
4.2 - Componente core (bra_core.c) 45
4.3 - Algoritmo de monitorização (bra_monitor.lua) 47
4.4 - Algoritmo de estimação (bra_estimator.lua) 48
4.5 - Algoritmo de adaptação (bra_adaptor.lua) 49
4.6 - Testes 50
4.6.1 - Algoritmo de estimação 50
4.6.2 - Algoritmo de monitorização 52
4.7 - Validação da solução - *Bit Rate Adaptation* vs *Constant Bit Rate* 54
4.7.1 - Análise objectiva 54
4.7.2 - Análise subjectiva 57

Capítulo 5 61

Conclusão 61
5.1 - Revisão do trabalho desenvolvido 61
5.2 - Principais dificuldades 62
5.3 - Possíveis melhorias 63
5.4 - Trabalho futuro 64

Referências 65

Lista de figuras

Figura 1-1 – Configuração genérica de um sistema de *streaming*. 2

Figura 1-2 – Ilustração da QoE e QoS. 4

Figura 2-1 – Ligação entre servidor e terminal, utilizando um protocolo de *streaming* tradicional (RTSP). 9

Figura 2-2 – Ligação entre servidor e terminal, utilizando um protocolo do tipo *download* progressivo (sobre HTTP). 10

Figura 2-3 – Ligação entre servidor e terminal, utilizando um protocolo do tipo *Streaming* Adaptativo. 11

Figura 2-4 – Troca de mensagens durante uma sessão utilizando o protocolo RTSP. 13

Figura 2-5 – Diagrama de estados de uma sessão RTSP..... 14

Figura 2-6 – Troca de relatórios entre cliente e servidor, utilizando o protocolo RTCP. 16

Figura 2-7 – Configuração básica de um sistema de transmissão com HTTP-Live Streaming. . 18

Figura 2-8 – Formato dos ficheiros Smooth Streaming..... 20

Figura 2-9 – Formato dos ficheiros *Wire* (fragmentos indicados na figura 2-8). 20

Figura 2-10 – Diagrama de estados para a detecção de largura de banda utilizando o protocolo Adobe Dynamic Streaming. 23

Figura 2-11 – Diagrama de estados da inicialização de uma sessão utilizando o protocolo Adobe Dynamic Streaming. 24

Figura 2-12 – Diagrama de estados da reprodução e monitorização de uma sessão utilizando o protocolo Adobe Dynamic Streaming..... 24

Figura 2-13 – Diagrama de estados do processo de aumento de qualidade de conteúdos utilizando o protocolo Adobe Dynamic Streaming..... 25

Figura 2-14 – Diagrama de estados do processo de redução de qualidade de conteúdos utilizando o protocolo Adobe Dynamic Streaming..... 26

Figura 2-15 – Diagrama de estados da transição entre conteúdos de diferentes qualidades utilizando o protocolo Adobe Dynamic Streaming..... 26

Figura 3-1 – Transmissão sobre TCP..... 34

Figura 3-2 – Variações do *jitter*. 38

Figura 4-1 – Arquitectura do módulo WSSBRA..... 42

Figura 4-2 – Diagrama de sequência do módulo WSSBRA..... 44

Figura 4-3 – Diagrama de sequência do componente Core do WSSBRA..... 45

Figura 4-4 – Diagrama de sequência do algoritmo monitor do WSSBRA. 47

Figura 4-5 – Diagrama de sequência do algoritmo de estimação do WSSBRA. 49

Figura 4-6 – Gráficos obtidos nos testes ao algoritmo de estimação, com erros médios: (a) 12,24 %; (b) 12,08%; (c) 14,60%; (d) 13,04%..... 51

Figura 4-7 – Gráficos obtidos nos testes ao algoritmo de monitorização. 53

Figura 4-8 – Sessões com *bit rate* constante: (a)300 kbit/s, (b)450 kbit/s e (c)600 kbit/s; Sessões com *bit rate* variável (limites máximos): (a)400 kbit/s, (b)550 kbit/s e (c)700 kbit/s..... 55

Figura 4-9 – Gráficos relativos aos vídeos avaliados subjectivamente. 57

Lista de tabelas

Tabela 2-1 — Soluções de *streaming*. 27

Tabela 4-1 — Relação causa-efeito para os algoritmos de monitorização e adaptação. 50

Tabela 4-2 — Resultados dos testes realizados ao módulo de adaptação. 56

Tabela 4-3 — Resultados objectivos para os vídeos utilizados na avaliação subjectiva. 58

Tabela 4-4 — Resultados da avaliação subjectiva realizada por 35 participantes. 58

Abreviaturas

AAC	*Advanced Audio Coding*
AAC-LC	*Advanced Audio Coding - Low Complexity profile*
AMF	*Action Message Format*
ANSI	*American National Standards Institute*
API	*Application Programming Interface*
AVC	*Advanced Video Coding*
BRA	*Bit Rate Adaptation*
CSV	*Comma Separated Values*
DSP	*Digital Signal Processor*
EDGE	*Enhanced Data rates for GSM Evolution*
FEUP	Faculdade de Engenharia da Universidade do Porto
FMS	*Flash Media Server*
GOP	*Group of Pictures*
GSM	*Groupe Spécial Mobile*
HE-AAC	*High-Efficiency - Advanced Audio Coding*
HSDPA	*High Speed Downlink Packet Access*
HTTP	*Hyper Text Transfer Protocol*
ICT	*Information and Communication Technologies*
IIS	*Internet Information Services*
IP	*Internet Protocol*
MOS	*Mean Opinion Score*
MPEG	*Moving Pictures Experts Group*
OS	*Operating System*
OSI	*Open Systems Interconnection*
PID	*Proporcional Integrative Derivative*
PT	Portugal Telecom
QoE	*Quality of Experience*
QoS	*Quality of Service*
RFC	*Request for Comments*
RTCP	*Real-time Transport Control Protocol*
RTMP	*Real-Time Messaging Protocol*
RTP	*Real-time Transfer Protocol*
RTSP	*Real-Time Streaming Protocol*
RTT	*Round Trip Time*
SMAN	*Session Manager*

SNMP	*Simple Network Management Protocol*
SWF	*Shockwave Flash*
TCP	*Transmission Control Protocol*
TS	*Transport Stream*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
VoD	*Video on Demand*
WLOG	*Windless Log*
WMA	*Windows Media Audio*
WOSX	*Windless Operative System Abstraction*
WSS	*Windless Streaming Server*
WSSBRA	*Windless Streaming Server - Bit Rate Adaptation*

Capítulo 1

Introdução

Actualmente, os operadores de televisão possuem um negócio de banda larga no qual estão acostumados a publicitar e vender conteúdos na televisão ou em monitores de computador. No entanto, estão cada vez mais a mostrar interesse nos ambientes móveis, começando a entrar no mercado através de operadoras móveis com redes virtuais ou ainda através da compra de espectro, convergindo assim em operadoras de serviços móveis. Isso significa que os operadores estão a explorar os serviços móveis, enquanto que as redes de televisão estão cada vez mais a atingir público através da *web*. Estas mudanças trazem consigo algumas dificuldades, como os utilizadores estão habituados a ver televisão com elevada qualidade, os seus padrões de aceitação são elevados. Esse facto aliado às menores capacidades das redes móveis em comparação com as fixas faz com que a reprodução de forma consistente da qualidade da televisão ou computador em terminais móveis seja uma tarefa difícil.

Com o aparecimento dos serviços/redes sociais como o Youtube, Google Videos ou Facebook, o tráfego de conteúdos multimédia está em grande expansão nas redes de banda larga fixas e móveis. No entanto, conteúdos criados para a televisão são raramente optimizados para transmissões móveis. De facto, os formatos são normalmente inadequados. Além disso, transmitir conteúdos de alta-definição e dimensões elevadas para terminais com capacidade de reprodução limitada (como são os telemóveis), leva a que a rede seja subaproveitada, surgindo a necessidade de adaptação dos conteúdos transmitidos.[1]

1.1 - Conteúdos multimédia

O aparecimento de novos dispositivos é habitualmente acompanhado não só pela disponibilização de novos serviços e aplicações, mas também pelo aumenta das expectativas dos consumidores. Com maiores ecrãs, os utilizadores cada vez mais esperam maiores e melhores resoluções. Estes dispositivos têm grande impacto na forma como os conteúdos têm de ser entregues pelas redes. O tamanho e qualidade do ecrã ou dispositivo define qual o *codec*, formato, e *bit rate* a usar para proporcionar a melhor experiência possível.

Actualmente, o aumento da qualidade da experiência fornecida pode levar a largura de banda utilizada ao seu limite. Os dispositivos cada vez são melhores, o que contribui para a necessidade de transferências mais eficazes, justificando a procura pela optimização da largura de banda.

Nos últimos tempos, tem-se verificado um aumento significativo na publicidade e venda dos *netbooks*. Estima-se que em 2009, a percentagem destes dispositivos no mercado global de computadores portáteis tenha sido de 17,2%, registando 26,3 milhões de compras[2]. Mais de metade de todos os *netbooks* será vendida por operadoras móveis ou fixas em *kits* juntamente com o serviço de Internet em 2014[3].

Além disso, as aplicações disponibilizadas são cada vez mais apelativas, sendo desenvolvidas para proporcionar uma fácil navegação, procura e compra de conteúdos. Basta ver o crescimento exponencial dos mercados de aplicações[4]. Toda esta procura e transferência de dados implicam elevados requisitos para a rede.

1.2 - Streaming

Referindo agora a forma como os conteúdos multimédia são entregues a utilizador, o *streaming* consiste na transmissão de dados – representada na **Figura 1-1**, quer de áudio quer de vídeo, de um servidor para um cliente que descodifica e reproduz os dados à medida que são recebidos tentanto preservar a relação temporal da fonte. Possibilita que o utilizador veja os conteúdos pretendidos imediatamente após um pequeno período de carregamento (*buffering*).

O *streaming* de conteúdos multimédia através da Internet foi popularizado pela Progressive Networks em 1995 com o seu formato proprietário RealAudio Streaming. Em 1997, na mesma altura em que alterou o seu nome para RealNetworks, esta empresa iniciou uma parceria com a Netscape com o intuito de desenvolver o que mais tarde se tornou o standard RTSP para *streaming*.

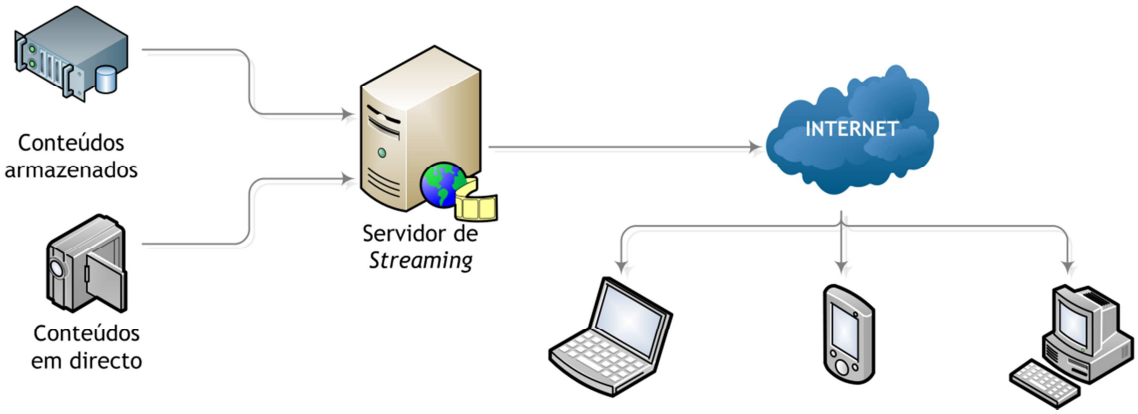

Figura 1-1 – Configuração genérica de um sistema de *streaming*.

Ao contrário da Internet, onde o *download* é ainda a forma mais popular de distribuição de conteúdos, nos aparelhos móveis a distribuição de conteúdos multimédia está a convergir rapidamente para o *streaming* (particularmente nas gravações em directo ou conteúdos televisivos), pois este apresenta algumas das vantagens quando comparado com a reprodução local após *download*:

- Não existirão dados gravados permanentemente no cliente final (ideal para equipamentos móveis onde a dimensão da memória é mais reduzida);
- O utilizador não necessita de esperar que todo o conteúdo seja recebido para iniciar a reprodução;
- Possibilita a reprodução de programas em directo;
- Uma grande vantagem para os fornecedores de serviços é o facto de não ser possível ao utilizador final reencaminhar ou enviar os conteúdos para outros utilizadores, e por isso os conteúdos podem ser cobrados de cada vez que são consumidos.

O *streaming* inicia-se com a codificação do ficheiro ou sinal a transmitir e, de seguida, é feita a divisão do sinal em pequenos pacotes que serão enviados sequencialmente através da Internet. Quando os pacotes chegam ao seu destino (o terminal do utilizador que pediu o serviço), são descomprimidos e reorganizados de forma a serem reproduzíveis pelo dispositivo do utilizador. Para manter a ilusão de reprodução contínua, os pacotes são introduzidos num *buffer* (como uma fila de espera), para que alguns deles já estejam no sistema do utilizador antes do início da reprodução. Enquanto os pacotes guardados são reproduzidos, outros pacotes são recebidos, e introduzidos na fila de espera para posterior reprodução. Contudo, quando a recepção de pacotes é muito lenta possivelmente devido a problemas de rede, o dispositivo do utilizador não tem pacotes para reproduzir e, enquanto aguarda a recepção dos dados em falta, suspende a reprodução, degradando a QoE do utilizador.

1.3 - QoE (Quality of Experience)

> *"A measure of user performance based on both objective and subjective measures of using an ICT [Information and Communication Technologies] service or product. (ETSI TR 102 643)"*

A anterior citação define QoE (Quality of Experience[5]) como uma medida do desempenho de um utilizador com base em parâmetros tanto objectivos como subjectivos na utilização de um produto. Na medição, devem ser incluídos todos os componentes do sistema que possam ter influência no serviço disponibilizado ao utilizador (cliente, terminal, rede, servidor, os próprios conteúdos, entre outros) e pode ainda ver-se influenciada tanto pelas expectativas dos utilizadores como pelo contexto em que é recolhida. Consequentemente, pode diferir bastante de utilizador para utilizador.

No caso desta dissertação, o conceito de QoE está directamente relaccionado com a QoS (Quality of Service), sendo sobre esta que a solução desenvolvida vai actuar.

1.3.1 - QoS (Quality of Service)

"Totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service. (ETSI TR 102 643)"

Como representado na **Figura 1-2**, o conceito de QoE está directamente relaccionado com o QoS, tornando-se difícil mencionar um sem falar no outro. Como a citação acima indica, em telecomunicações, o termo QoS está aliado a um conjunto de características que proporcionam ao respectivo serviço a capacidade de satisfazer as necessidades explícitas e implícitas do utilizador final.

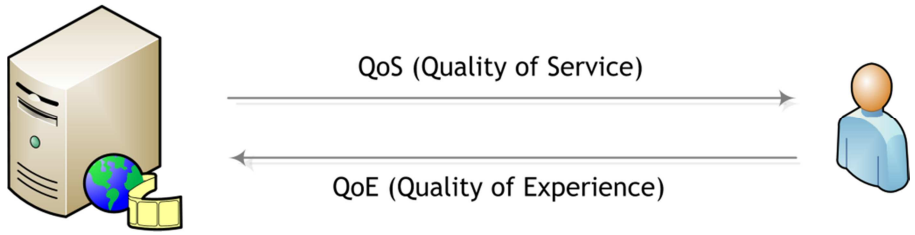

Figura 1-2 – Ilustração da QoE e QoS.

No âmbito desta dissertação, o conceito de QoS tem especial importância porque, apesar de no tema ser referida a qualidade de experiência, na verdade a solução a desenvolver tem como objectivo diminuir as perdas resultantes da transmissão através da rede móvel. Valores como o número de pacotes perdidos, ou a quantidade de dados transmitidos sobre um determinado período de tempo são normalmente medidas inseridas na QoS.

Disponibilizar QoS basicamente significa proporcionar garantias de transmissão para certos fluxos de dados. A garantia de transmissão pode ser expressa como a combinação de alguns dos seguintes parâmetros[6]:

- **Atraso** - o tempo que um pacote demora a percorrer a rede;
- **Variação do atraso (*jitter*)** - a variação no atraso entre dois pacotes consecutivos;
- **Largura de banda** - a taxa de transferência máxima disponível para a transmissão;
- **Confiabilidade** - a taxa de pacotes perdidos, corrupção e perda de sincronismo.

Em sessões de *streaming*, a variação destes valores tem implicações directas na QoE fornecida ao utilizador. Isto é, se o valor do atraso aumentar para valores acima da média, a reprodução de conteúdos será eventualmente interrompida para que a aplicação cliente receba os dados necessários para apresentar ao utilizador. Caso a confiabilidade diminua, ou seja, sejam perdidos ou corrompidos pacotes, será visível uma degradação na qualidade do vídeo reproduzido, visto que ocorrerão falhas no meio das imagens.

Assim, optimizar estes valores significa aumentar a QoS o que representa uma melhoria na QoE fornecida ao utilizador.

1.4 - Motivação

À medida que os pedidos por conteúdos multimédia através de computadores, televisão ou terminais móveis aumentam drasticamente, também os níveis de exigência por uma QoE satisfatória por parte do utilizador se elevam. Infelizmente, a largura de banda limitada, o mercado de dispositivos e reprodutores multimédia muito fragmentado e em constante actualização, e ainda a imprevisibilidade do tráfego e condições de rede dificultam aos operadores o envio de conteúdos multimédia com qualidade correspondente às expectativas dos utilizadores.

De acordo com a previsão para a quantidade de tráfego móvel da Cisco VNI (Visual Networking Index), o consumo de conteúdos áudiovisuais irá duplicar de ano para ano até 2014. Nesse momento, este tipo de conteúdos vai ser responsável por cerca de 60% de todo o tráfego na rede[7]. Em 2013, 2.1 mil milhões de utilizadores com subscrição irá pagar e descarregar *mobile TV*, conteúdos gerados por utilizadores (por exemplo, do Youtube), e música através de dispositivos móveis, representando um crescimento sete vezes superior nas transacções efectuadas em 2008[8]. Além disso, as receitas obtidas a partir dos serviços de *streaming* de conteúdos multimédia deverão crescer nos próximos anos, aumentando de 2.5 mil milhões em 2009 para aproximadamente 5.5 mil milhões em 2013[9].

Todas estas tendências e oportunidades representam desafios únicos para os operadores de redes móveis e de banda larga. Cada vez mais utilizadores descarregam conteúdos multimédia; conteúdos gerados por utilizadores apresentam taxas de crescimento cada vez maiores; no entanto, os operadores possuem limitadas capacidades monetárias e físicas para gerir toda esta expansão. Enquanto que a largura de banda requisitada pelos utilizadores duplicará todos os anos, os operadores de rede não terão possibilidade de aumentar a sua capacidade a essa mesma taxa. O que significa que a procura por soluções que permitam aos operadores acompanhar o crescimento do mercado sem alterações drásticas nas suas redes serão uma realidade. Essas soluções passam por novas formas de transmissão e controlo dos conteúdos enviados para os utilizadores. Entre elas, a adaptação dos conteúdos à qualidade da rede de cada utilizador não só representa uma solução orientada ao cliente, como permite a optimização do consumo de recursos da rede dos operadores.

1.5 - Objectivos

Neste documento pretende-se apresentar uma solução para a implementação de um módulo que visa melhorar a QoE do utilizador final numa transmissão e reprodução de conteúdos multimédia. Este serviço de rede assume uma grande importância para empresas na área das telecomunicações uma vez que tem como objectivo aumentar a satisfação de clientes, optimizando a utilização de recursos de rede.

A solução a desenvolver deverá ser compatível com diferentes tipos de terminais, desde *smartphones* aos mais antigos *feature phones*, passando por computadores portáteis, sendo que os únicos requisitos são a conectividade com o servidor de conteúdos multimédia que

utiliza a solução, e ainda a compatibilidade com os protocolos utilizados no módulo desenvolvido.

O principal objectivo da solução a desenvolver é utilizar as ferramentas disponibilizadas pelo servidor de *streaming* da PT Inovação para adaptar a transmissão dos conteúdos às condições de rede de cada utilizador, sempre com o intuito de melhorar a QoE deste.

Para além das melhorias subjectivas esperadas com a aplicação da solução desenvolvida, também devem ser consideradas melhorias objectivas que possam melhorar a QoE dos utilizadores. Por exemplo, se a utilização dos recursos de rede for optimizada, os utilizadores possuirão melhor qualidade de ligação, podendo assim receber conteúdos com maior qualidade originando consequentemente maior satisfação.

A solução a desenvolver deve ainda ser de fácil configuração, adaptação e integração em diferentes sistemas. Para isso devem ser utilizadas linguagens de programação adequadas e de conhecimento geral.

1.6 - Organização do documento

O documento está organizado em cinco capítulos. No cômputo geral, este documento pretende realizar uma introdução aos conceitos relativos à transmissão de conteúdos multimédia (i.e. áudio, vídeo), identificar e descrever os diferentes serviços e aplicações existentes, apresentar a solução desenvolvida com a respectiva prova de conceito e por fim expor as conclusões do trabalho realizado.

No primeiro capítulo é realizado um enquadramento do projecto na actualidade das tecnologias de informação, seguida de uma breve descrição dos principais conceitos relacionados com transmissão de conteúdos multimédia ao longo da *web* e respectiva qualidade. Após os conceitos iniciais, faz-se uma descrição da motivação e dos objectivos inerentes à realização deste trabalho.

No segundo capítulo é feito um levantamento do estado da arte com a descrição dos diferentes tipos de abordagens actualmente seguidos por diferentes serviços existentes. Posteriormente é feito o levantamento e enquadramento dos diferentes serviços e protocolos de transmissão apresentando as suas principais características. De seguida, é feita uma comparação entre as diferentes soluções encontradas, terminando com a justificação da necessidade de uma nova solução.

No terceiro capítulo são apresentadas as justificações para a escolha de protocolos a utilizar no módulo desenvolvido. De seguida são descritos conceitos importantes que foram considerados na implementação da solução. Finalmente, são apresentadas as tecnologias que servem de suporte ao funcionamento do módulo, juntamente com o seu enquadramento no âmbito da dissertação.

O quarto capítulo apresenta a solução proposta para a implementação de um módulo de adaptação com a descrição da arquitectura utilizada, e as funcionalidades e objectivos de

cada componente. Em seguida, são descritas com detalhe todas as implementações efectuadas. Posteriormente, é apresentada uma secção onde são descritos todos os testes realizados aos componentes desenvolvidos, juntamente com os resultados da aplicação da solução e respectiva análise. Finalmente é apresentada a validação da solução desenvolvida, que inclui uma avaliação subjectiva por parte de diferentes utilizadores.

Por fim, no quinto capítulo é feito um balanço sobre todo trabalho realizado, com destaque para os objectivos que foram atingidos e o que poderá ser melhorado. São ainda expostas todas as dificuldades sentidas ao longo do trabalho e as lições aprendidas. O capítulo termina com uma abordagem ao trabalho futuro a ser realizado.

Capítulo 2

Estado da Arte

Este capítulo apresenta o levantamento do estado da arte, efectuado antes do início do desenvolvimento do módulo de adaptação. São expostas as diferentes abordagens para a transmissão de conteúdos multimédia, descritos os vários serviços e protocolos de *streaming* existentes, o seu modo de funcionamento e é ainda efectuada uma comparação entre os mesmos. Finalmente é justificada a necessidade da solução.

2.1 - Multimedia Streaming

2.1.1 - Diferentes Abordagens

Actualmente, na entrega de conteúdos multimédia através da Internet são utilizados três tipos genéricos de abordagens de *streaming*: *streaming* tradicional, *download* progressivo, e *streaming* adaptativo sobre HTTP[10].

***Streaming* Tradicional**

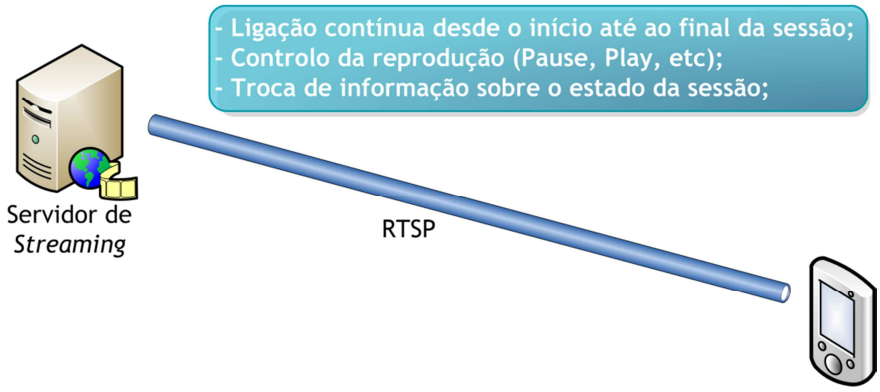

Figura 2-1 – Ligação entre servidor e terminal, utilizando um protocolo de *streaming* tradicional (RTSP). Figura adaptada de [10].

Utilizando o Real Time Streaming Protocol como referência, uma das principais características desta abordagem e que a distingue das outras, é o facto de possuír sessões. Estas são iniciadas no momento em que é estabelecida a ligação, e terminam quando a transmissão dos conteúdos multimédia se completada ou interrompida. Desta forma, durante toda a sessão, existe apenas uma transmissão contínua de um fluxo de dados, ao contrário dos outros tipos em que existem vários segmentos e respectivas transmissões para cada conteúdo multimédia/programa. Como existem sessões, estas têm de ser monitorizadas; assim sendo, nesta abordagem existe troca de mensagens acerca da sessão actual. Por esse motivo, os serviços que seguem esta abordagem são *stateful*, ou seja, os intervenientes na transmissão mantêm informação de estado sobre a sessão em curso. No entanto, esta monitorização não fornece qualquer informação sobre as condições da rede. Outra característica deste tipo de abordagem é o controlo da transmissão por parte do cliente, através do envio de mensagens pré-definidas como *Play* ou *Pause*.

Apesar de não serem transmitidos segmentos dos conteúdos/programas, a informação transmitida é codificada em pacotes de pequeno comprimento. Estes pacotes, no caso do RTSP, protocolo admitido como exemplo, podem ser codificados sobre UDP (User Datagram Protocol) ou TCP (Transmission Control Protocol). Caso sejam codificados em UDP, e porque este protocolo utiliza portas fora do comum, muitos firewalls e *proxies* bloqueiam os pacotes, comprometendo a qualidade do serviço. Esta é uma das grandes desvantagens deste tipo de protocolos, e levou mesmo à necessidade de criação de novas abordagens melhor adaptadas à Internet. As soluções adoptadas basearam-se na utilização do protocolo HTTP, amplamente utilizado e suportado na Internet.

***Download* Progressivo**

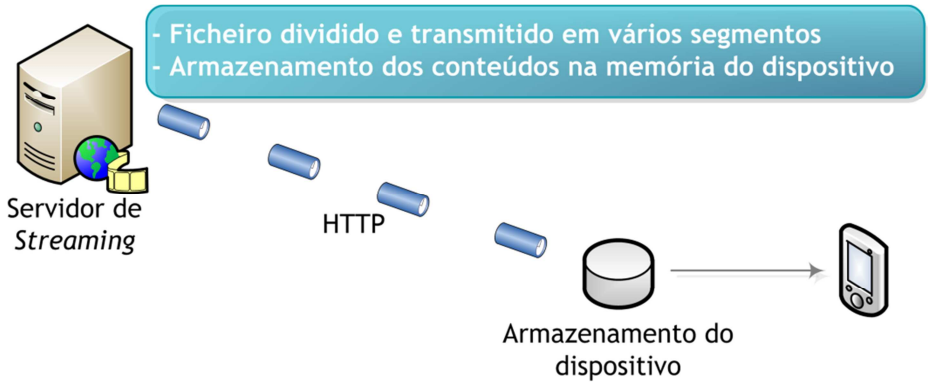

Figura 2-2 – Ligação entre servidor e terminal, utilizando um protocolo do tipo *download* progressivo (sobre HTTP). Figura adaptada de [10].

Esta abordagem consiste no *download* de ficheiros disponibilizados por um servidor *web* HTTP. Assim sendo, é suportado pela maioria das plataformas e aplicações de reprodução de conteúdos multimédia, tais como Adobe Flash, Silverlight, ou ainda Windows Media Player. O termo "Progressivo" enquadra-se neste tipo de abordagem uma vez que a maioria das aplicações de reprodução do lado do cliente permite que o conteúdo seja reproduzido enquanto o *download* ainda está a decorrer, ou seja, antes de o ficheiro estar completo. Clientes com suporte para HTTP 1.1 podem também escolher visualizar uma parte do conteúdo que ainda não esteja disponível no cliente. Isto é possível através da troca de mensagens entre o servidor e cliente, que envia pedidos denominados "*Byte Range*

Requests". Esta é a solução adoptada por vários *websites* de partilha de vídeos, como o Youtube, Vimeo, ou Myspace, que utilizam quase exclusivamente *download* progressivo.

Ao contrário dos servidores de *streaming* normais que raramente enviam conteúdos com mais de 10 segundos de uma vez só, os servidores *web* HTTP não param a transmissão dos dados até o *download* estar concluído. Se o utilizador suspender a reprodução de um conteúdo que está a ser transferido através de *download* progressivo e esperar um período de tempo indeterminado, todo o vídeo poderá eventualmente ser transferido para o cliente, possibilitando a reprodução completa sem as habituais interrupções. Uma desvantagem desta funcionalidade é que, se o utilizador esperar que todo o conteúdo seja transferido e, passado um curto período de tempo fechar o vídeo, tanto ele como o fornecedor dos conteúdos terão desperdiçado largura de banda que, no caso de ambientes móveis, é recurso por vezes bastante limitado.

***Streaming* Adaptativo**

Figura 2-3 – Ligação entre servidor e terminal, utilizando um protocolo do tipo *Streaming* Adaptativo. Figura adaptada de [10].

O *streaming* adaptativo consiste num método de transmissão híbrido que tem funcionamento semelhante ao *streaming* normal, mas baseando-se no *download* progressivo sobre HTTP. Este tipo de abordagem utiliza o HTTP como protocolo de transporte. No entanto, em vez de transmitir apenas um único fluxo de dados completo, transmite muitos segmentos, os quais, quando reproduzidos ordenadamente, formam o vídeo/conteúdo completo.

Numa implementação típica de *streaming* adaptativo, o conteúdo multimédia original é dividido em muitos segmentos, os quais, normalmente, têm duração curta (dependendo do protocolo utilizado, esta duração normalmente varia entre 2 e 10 segundos). Habitualmente, esta duração está directamente ligada com o tamanho do GOP (Group of Pictures) [1], ou seja, cada imagem I (independente das B e P) corresponde tanto ao fim de um segmento como ao

[1] Grupo de imagens de um segmento de vídeo codificado segundo as normas MPEG. Contém três tipos de imagem: I-independentes, P-diferença entre a actual e a imagem I ou P anterior, B-diferença entre a actual e a imagem I ou P anterior e/ou posterior.

início do seguinte. Desta forma, na altura da descodificação não existem dependências entre diferentes segmentos do vídeo, ou seja, para um segmento ser descodificado, apenas é necessário que todo ele tenha sido recebido pelo descodificador.

Os segmentos de vídeo codificados são guardados e disponibilizados por um servidor *web* HTTP. A aplicação cliente pede ao servidor os segmentos, e descarrega-os utilizando *download* progressivo. À medida que os segmentos são recebidos pelo cliente, este ordena-os linearmente, e faz a sua reprodução. Isto só é possível porque os segmentos são codificados cuidadosamente, sem que exista espaçamento ou sobreposição entre os segmentos, possibilitando que, ao serem reproduzidos ordenadamente, formem um fluxo contínuo.

Anteriormente foi explicado de que forma os serviços que seguem esta abordagem se enquadram no *streaming*; resta dizer em que consiste a adaptabilidade indicada no seu nome. Esta propriedade é aplicável quando, para o mesmo conteúdo de áudio ou vídeo, existem múltiplas codificações com *bit rates* distintos, ou seja, com qualidade diferente. A existência desta propriedade implica que sejam utilizados vários débitos binários na codificação do conteúdo, dando origem a ficheiros com tamanhos diferentes ou a fluxos binários com diferentes débitos. Logicamente, tamanhos superiores implicam maior largura de banda disponível para que os conteúdos sejam transmitidos, num tempo real ou maior que real que não conduza os *buffers* do receptor a *overflow*. Em ambientes onde a largura de banda não permite esta situação, revela-se necessária uma solução que possibilite continuar a visualizar o conteúdo, mesmo que este tenha qualidade inferior. Assim, como existem ficheiros ou fluxos com tamanhos e qualidades diferentes, a aplicação cliente poderá optar por diminuir a qualidade do ficheiro favorecendo a fluidez da reprodução. Da mesma forma que, se detectar que tem capacidade para receber vídeos com qualidades superiores, poderá pedir os mesmos ao servidor.

O *streaming* adaptativo oferece aos fornecedores de conteúdos vantagens relativamente ao *streaming* tradicional, tais como:

- A implementação é mais barata, pois pode usar caches/proxies HTTP genéricos e não necessita de servidores especializados;
- Maior escalabilidade e alcance, uma vez que se adapta dinamicamente a más condições de rede;
- Possibilita que cada utilizador receba os conteúdos com qualidade adequada ao estado da sua conexão e, dessa forma, não são prejudicados clientes que possuam melhor qualidade de ligação.

2.1.2 - RTSP (Real-Time Streaming Protocol)

O RTSP[11] é um protocolo ao nível da aplicação que fornece controlo sobre a entrega de dados em tempo-real. Este protocolo tem por objectivo controlar múltiplas sessões de transmissão de dados, possibilitar a escolha do protocolo de transporte, tal como UDP, *multicast* UDP ou TCP.

Visto que o objectivo principal do RTSP consiste no estabelecimento e controlo de sessões através da troca de mensagens entre servidor e um dado cliente e não propriamente a transmissão dos conteúdos em si, este protocolo pode ser visto como um "controlo remoto da rede" para servidores multimédia. Algumas das mensagens trocadas são referentes à reprodução do conteúdo, como por exemplo, os comandos *Play*, *Pause* ou *Teardown* o qual indica ao servidor que o cliente pretende terminar a sessão e consequentemente a transmissão. Na **Figura 2-4** está representada a troca de mensagens durante uma sessão RTSP, na qual é possível confirmar as características anteriormente descritas.

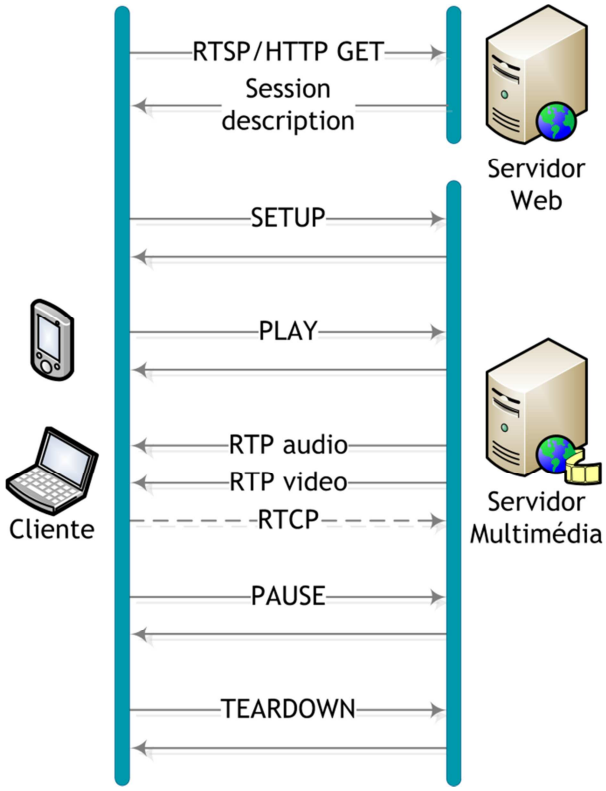

Figura 2-4 – Troca de mensagens durante uma sessão utilizando o protocolo RTSP.

Uma das grandes desvantagens deste protocolo é o facto das suas transmissões serem muitas vezes bloqueadas por *routers* e *firewalls*. Esta situação acontece, pois a transmissão utilizando RTP é maioritariamente efectuada sobre UDP que utiliza portas não muito comuns. Como os *routers* ou *firewalls* não reconhecem este tráfego bloqueiam-o por motivos de segurança. Apesar de este protocolo ser um dos desenvolvidos para transmissão de conteúdos multimédia, de facto o protocolo mais utilizado na Internet é HTTP e assim, a transmissão sobre HTTP foi optimizada relativamente a outros tipos([12], capítulo 4 – Compatibilidade).

Algumas das principais características do protocolo RTSP:

- Expansível;
- Independente do protocolo de transporte (UDP ou TCP);
- Compatível com multi-servidores;
- Permite o controlo de dispositivos de gravação;
- Separação entre as transmissões de controlo e inicialização de ligação;
- Negociação do tipo de transporte.

Funcionamento

No início de cada sessão RTSP, é disponibilizada uma lista denominada "*Presentation*", que contém os conteúdos disponíveis para o cliente. Existe ainda o campo *presentation description* que contém informação sobre uma ou mais *streams*, por exemplo, sobre os *codecs* utilizados, endereços de rede (no caso de *multicast*) e informação sobre o conteúdo. Este ficheiro pode ser obtido pelo cliente através de HTTP ou outros meios como email, e não tem necessariamente de estar armazenado no servidor multimédia.

Através deste ficheiro, o cliente pode escolher quais os conteúdos mais adequados para o seu caso. Cada *stream* controlável individualmente por RTSP está identificada por um RTSP URL (Uniform Resource Locator), que indica o servidor responsável pela *stream*. As *streams* multimédia podem estar armazenadas em servidores diferentes, por exemplo, conteúdos de áudio/vídeo podem estar divididos por diferentes servidores para diminuir a carga sobre os mesmos.

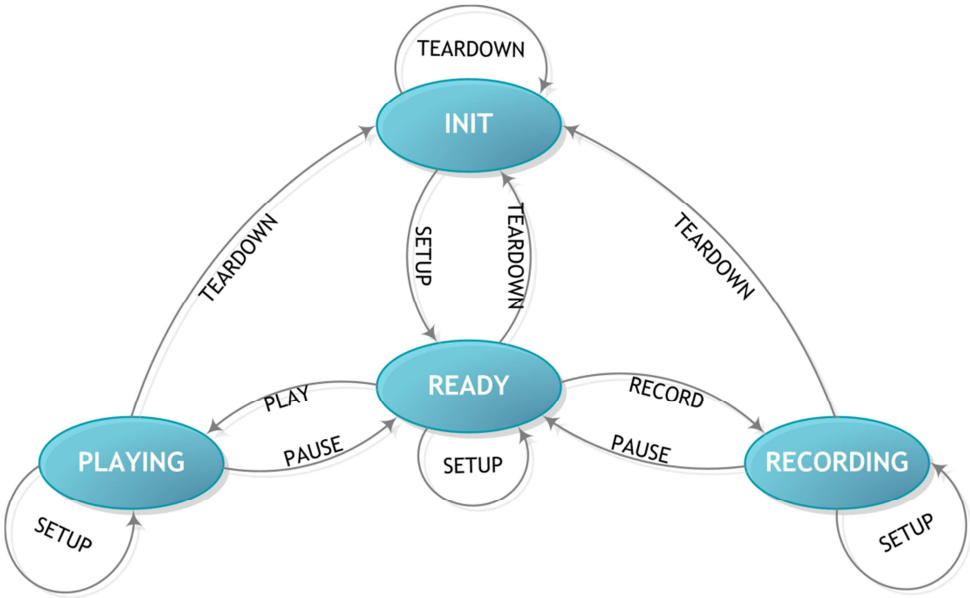

Figura 2-5 – Diagrama de estados de uma sessão RTSP.

Embora existam outros, os comandos básicos RTSP enviados do cliente para o servidor (representados na **Figura 2-5**), são os seguintes:

- ***SETUP***: despoleta a alocação de recursos pelo servidor para uma *stream*, e inicia a sessão RTSP;
- ***PLAY*** e ***RECORD***: relativos a um conteúdo anteriormente seleccionado através do comando *SETUP*;
- ***PAUSE***: pára temporariamente a *stream* sem libertar os recursos alocados;
- ***TEARDOWN***: liberta os recursos alocados à *stream*, a sessão RTSP é terminada.

Como referido anteriormente, as principais funções do protocolo RTSP são o controlo remoto de servidores multimédia, e a inicialização e manutenção de sessões. Este protocolo não tem interferência no transporte dos dados, deixando essas funções para outros protocolos, normalmente o RTP.

RTP (Real-time Transport Protocol)

O RTP[13] é o protocolo padrão da Internet para o transporte de dados, áudio ou vídeo, em tempo-real. É um protocolo ao nível da aplicação e possibilita a transmissão de dados de forma transparente entre terminais. Pode ser utilizado em *media-on-demand* ou ainda em serviços interactivos como telefonia através da Internet. É constituído por duas partes essenciais, dados e controlo. O último é designado Real-time Transport Control Protocol (RTCP).

RTCP (Real-time Transport Control Protocol)

O protocolo RTCP ([13], secção 6) tem especial importância para esta dissertação uma vez que tem como função estabelecer um canal de comunicação entre a aplicação cliente e o servidor, onde são trocadas mensagens de controlo com parâmetros relativos à transmissão e recepção de dados. Através destes parâmetros, é possível obter algumas estimativas que permitem tomar decisões sobre a necessidade de adaptar os conteúdos transmitidos e assim controlar a degradação da qualidade do vídeo transmitido.

Este protocolo baseia-se no envio periódico de pacotes de controlo para todos os participantes de uma sessão de *streaming*, utilizando o mesmo mecanismo de distribuição que os pacotes de dados.

A comunicação (representada na **Figura 2-6**) é feita utilizando dois tipos de mensagens:

- **RE**: Relatório de Emissor que contém estatísticas da transmissão e recepção de dados geradas por emissores activos;
- **RR**: Relatório de Receptor que contém estatísticas da recepção de dados geradas por participantes que não são emissores activos, ou combinados com RE para emissores activos que informem sobre mais de 31 fontes.

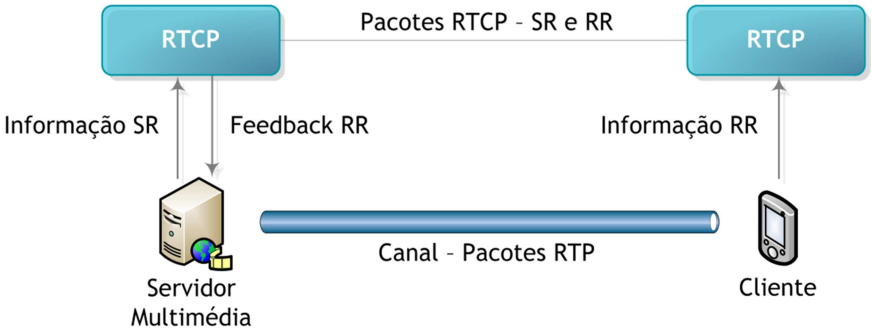

Figura 2-6 – Troca de relatórios entre cliente e servidor, utilizando o protocolo RTCP.

Nos relatórios do receptor são enviadas informações sobre a recepção dos dados. Algumas das informações contidas nos relatórios são:

- Número cumulativo de pacotes RTP recebidos pelo cliente;
- Variação estatística do tempo de recepção dos pacotes RTP;
- Número de pacotes perdidos;
- Número de pacotes esperados.

Alguns dos valores relevantes que podemos obter a partir dos relatórios RTCP são:

- Número de pacotes perdidos entre a recepção de dois relatórios consecutivos – diferença entre os dois valores do número de pacotes perdidos;
- Número de pacotes esperados durante um intervalo – diferença entre os números de sequência recebidos;
- Taxa de pacotes perdidos no intervalo – razão entre dois valores acima os anteriores;
- Taxa de perdas por segundo – quociente entre a taxa de pacotes perdidos e a diferença entre *timestamps*;
- Variação inter-recepção – fornece uma estimativa imediata relativa ao congestionamento da rede.

2.1.3 - HTTP-Live Streaming

Durante a última década, a Apple tem vindo a utilizar uma solução chamada QuickTime Streaming Server, a qual utiliza um servidor RTSP para transmitir conteúdos multimédia previamente gravados ou em directo para os seus clientes. Contudo, apesar de oferecer streaming gratuito, o Quicktime Streaming Server não atingiu as expectativas inicialmente criadas[14].

Uma das grandes razões para o rendimento abaixo do esperado é o facto do tráfego transmitido com RTSP ser bloqueado por muitos *firewalls* e *routers*, dificultando a entrega de *streams* com sucesso. Esta situação seria ainda pior caso se aplicasse esta solução em ambientes móveis, pois é natural que os terminais alternem entre redes móveis e Wi-Fi, utilizando por isso diferentes elementos de rede, com diferentes configurações de segurança.

Tendo em vista a resolução do problema de compatibilidade do protocolo RTSP com os *firewalls*, a Apple decidiu adicionar uma opção ao QuickTime Streaming Server. Esta consistiu na inserção do tráfego RTSP em pacotes HTTP que eram idênticos aos transmitidos habitualmente na Internet e, dessa forma, não eram bloqueados pela grande maioria dos *firewalls*. Esta solução tinha como consequência o aumento da informação enviada no cabeçalho (obrigatória para possibilitar a descodificação do sinal), resultando no aumento da largura de banda necessária para a transmissão do sinal. Assim sendo, e porque a largura de banda em ambientes móveis é um recurso relativamente limitado, revelou-se necessário criar de um novo protocolo concebido a pensar nesta situação. Esse protocolo foi denominado HTTP-Live Streaming.

Essencialmente, o conteúdo a transmitir é codificado numa MPEG Transport Stream[15] e posteriormente dividido em segmentos com duração de, aproximadamente, 10 segundos. Em vez de pedir um fluxo (*stream*) contínuo com novos dados codificados sobre RTSP, este novo protocolo simplesmente pede o primeiro par de segmentos, pedindo os restantes de acordo com a necessidade e codificando-os sobre HTTP. Desta forma a compatibilidade com *firewalls* é bastante melhorada, e não são necessários servidores com características específicas, pois qualquer servidor *web* comum consegue transmitir os segmentos de vídeo.

Outra das funcionalidades do HTTP-Live Streaming consiste na possibilidade do servidor conter várias versões dos segmentos de vídeo em diferentes formatos. Desta forma, os terminais móveis com maior largura de banda disponível podem negociar versões com qualidade superior relativamente aos terminais com menor largura de banda disponível, (por exemplo, Wi-Fi ou EDGE - Enhanced Datarates for GSM Evolution - respectivamente). Com esta funcionalidade, é oferecida ao utilizador a melhor qualidade de vídeo possível para a largura de banda disponível em cada caso, sendo que esta é continuamente optimizada à medida que novos segmentos são pedidos.

Visto que um dos objectivos da Apple na criação deste protocolo era a sua compatibilidade com equipamentos existentes, este protocolo foi desenvolvido para funcionar com qualquer reprodutor de multimédia, independentemente da plataforma utilizada no dispositivo móvel. Uma grande vantagem desta implementação reside no facto de não ser necessário instalar programas extra, sendo que os editores de conteúdos poderão publicá-los numa página Internet comum, à qual os dispositivos podem aceder como se de um computador normal se tratasse.

Funcionamento

Para perceber o funcionamento do HTTP-Live Streaming foi analisada com pormenor a *Technical Overview*[16] do mesmo, e ainda o *draft*[17] disponibilizado pela Apple.

Como podemos verificar na **Figura 2-7**, o processo de transmissão inicia-se com a disponibilização dos conteúdos multimédia ao servidor. Estes podem ser originários de gravações em directo, ou anteriormente gravados e armazenados. Depois de recebidos pelo

servidor, os conteúdos são, normalmente, codificados dentro de uma MPEG-2 Transport Stream, utilizando dispositivos de codificação comuns. Seguidamente, a *stream* MPEG-2 é dividida em segmentos, e guardada como um conjunto de um ou mais ficheiros multimédia com extensão ".ts". Normalmente, para a execução deste passo, é utilizada uma aplicação como a da Apple, designada Stream Segmenter.

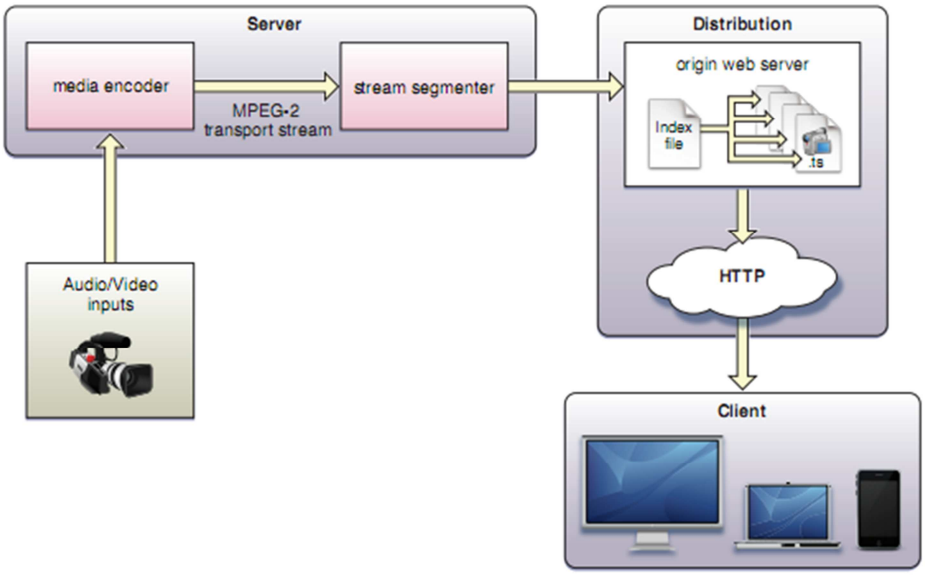

Figura 2-7 – Configuração básica de um sistema de transmissão com HTTP-Live Streaming. Figura retirada de [16].

O segmentador gera também um ficheiro *index* em formato ".M3U8", que contém a lista dos segmentos disponíveis para reprodução. Este ficheiro contém ainda metadados. Os conteúdos deste ficheiro são consultados pelos clientes, que posteriormente requisitam os ficheiros indexados sequencialmente.

No ficheiro *index* podem ainda estar indicadas *streams* alternativas para os mesmos conteúdos. São estas referências que possibilitam a adaptação dinâmica da qualidade dos conteúdos reproduzidos. Durante a transmissão, a aplicação cliente utiliza uma heurística para determinar a necessidade de alteração das características do conteúdo, ou seja, maior ou menor qualidade consoante a qualidade da ligação.

O servidor deve garantir que estão presentes na lista pelo menos três segmentos a cada momento, excepto se estiver indicado o comando EXT-X-ENDLIST que indica o final do conteúdo; caso contrário não é possível iniciar uma transmissão. À medida que um segmento é reproduzido, deverá existir outro pronto a reproduzir gravado no dispositivo, e ainda um terceiro a ser recebido. À medida que vão sendo transmitidos, os segmentos são apagados da *playlist* pela mesma ordem porque foram introduzidos. No entanto, apenas a informação nas listas é removida imediatamente, visto que o segmento deverá ficar disponível durante mais algum tempo, de forma a não prejudicar o último cliente a visualizá-lo.

Até que seja detectado o comando que indica o final de um dado conteúdo, o cliente deve actualizar o ficheiro *index*, sendo que o tempo entre actualizações está limitado por valores relacionados com a duração dos segmentos.

Este protocolo possui também tipos de sessões que ajudam o cliente a identificar qual a proveniência dos conteúdos que está a reproduzir. Para conteúdos transmitidos em directo, deverá ser usada uma sessão do tipo "*ongoing*". Esta é identificada quando não existe um comando de final de transmissão. Neste caso, o cliente não consegue aceder aos dois últimos ficheiros da lista, e quando inicia a transmissão do terceiro ficheiro para o final, faz a actualização da lista.

Como sempre que um cliente actualiza o ficheiro *index*, um novo ficheiro é transmitido pelo servidor, pode ocorrer sobrecarga da rede, caso a duração dos segmentos, (que define o tempo entre actualizações), seja mal dimensionada. Caso seja demasiado curta, os clientes solicitarão actualizações demasiadas vezes; caso seja demasiado longa, o ficheiro *index* conterá muitos segmentos e, por isso, tornar-se-á relativamente grande.

Neste protocolo, a detecção da largura de banda disponível é feita inteiramente pelo cliente, e pode ser calculada através do tempo de *download* dos ficheiros enviados pelo servidor. Também pode ser feita através da monitorização do espaço livre no *buffer*. Se existir pouca informação neste, devemos diminuir a qualidade, para que a transmissão seja mais rápida, evitando que faltem conteúdos para a reprodução. Da mesma forma, se o *buffer* estiver a encher, podemos aumentar a qualidade do conteúdo.

Depois de recolher dados sobre a qualidade de rede disponível, o cliente apenas tem de pedir ao servidor os conteúdos presentes nas *playlists* que mais se adequam à sua situação.

2.1.4 - Smooth Streaming

Em Outubro de 2008, a Microsoft anunciou o lançamento de uma nova extensão de *streaming* adaptativo sobre HTTP para o IIS 7.0 (Internet Information Services 7.0), denominado Smooth Streaming[18].

Esta extensão detecta dinamicamente a largura de banda e estado do processador de que o utilizador dispõe e, de forma transparente para este, faz a troca em tempo próximo do real da qualidade do vídeo recebido pela aplicação cliente.

Uma das grandes vantagens do Smooth Streaming é a forma como são guardados os ficheiros que contêm os conteúdos a transmitir. Ao contrário de outras soluções, esta não divide os ficheiros originais fisicamente, ou seja, no servidor, em vez de estarem guardados pequenos ficheiros para cada conteúdo/programa, estão guardados os ficheiros/programas completos. Nesta solução, a divisão é feita virtualmente de acordo com a necessidade dos clientes, melhorando significativamente o processo de gestão de ficheiros. De realçar que continuam a existir diferentes ficheiros com qualidade distinta para o mesmo conteúdo de modo a possibilitar a adaptação da qualidade do vídeo ao estado da rede.

Funcionamento

Esta solução de *streaming* sobre HTTP desenvolvida pela Microsoft, ao contrário das desenvolvidas por esta empresa na última década, utiliza o formato de ficheiros MP4. Na realidade, nesta solução existem duas partes para o formato utilizado, o formato *wire* e o

disk file. Em Smooth Streaming, um vídeo é gravado e armazenado nos servidores num só ficheiro com todo o conteúdo (um ficheiro por *bit rate*), mas é transmitido para a aplicação cliente como um conjunto de pequenos segmentos. Assim sendo, o formato "*wire*", representado na **Figura 2-9** define a estrutura dos segmentos enviados pelo IIS para o cliente, enquanto o formato "*file*", **Figura 2-8**, define a estrutura dos ficheiros gravados nos servidores. Como a especificação MP4 possibilita que os ficheiros respectivos sejam interiormente organizados como uma série de segmentos, o formato *wire* é apenas um subconjunto do formato MP4.

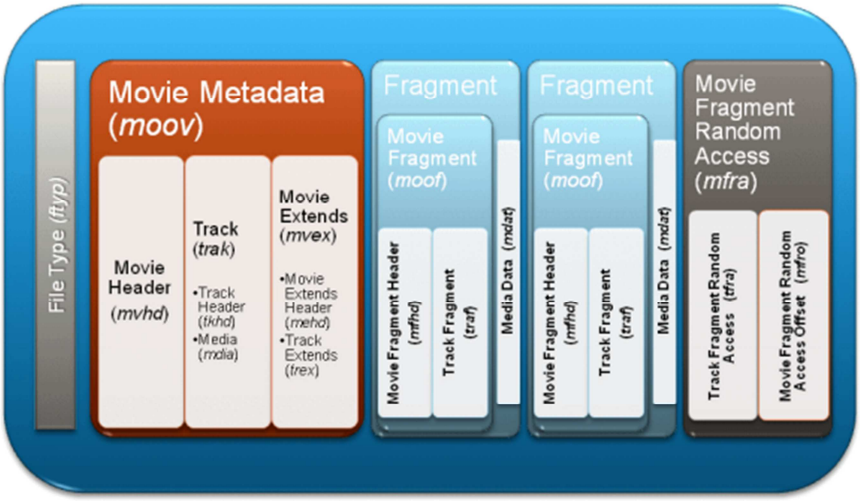

Figura 2-8 – Formato dos ficheiros Smooth Streaming. Figura retirada de [18].

O ficheiro inicia-se com metadados que normalmente descrevem o ficheiro; contudo, a maioria da informação está realmente contida nas *fragment boxes* que transportam metadados mais específicos ao nível dos fragmentos, e ainda media data que são os conteúdos propriamente ditos. No final do ficheiro, está um campo que permite a procura fácil e precisa dentro do ficheiro.

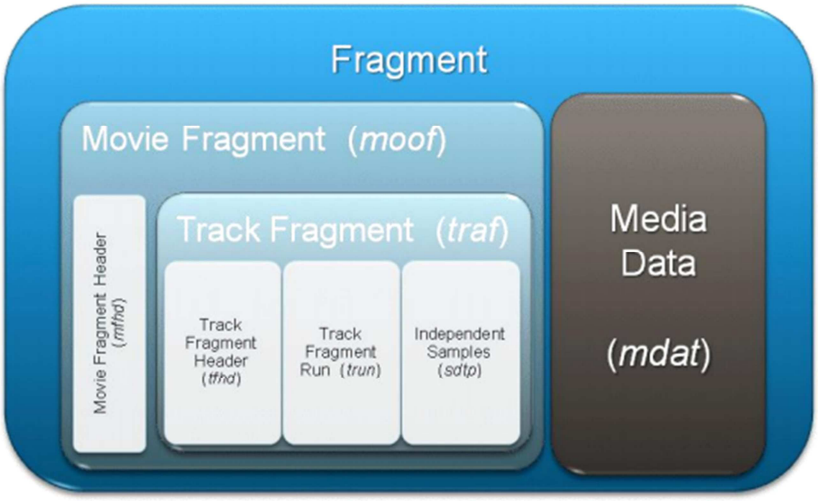

Figura 2-9 – Formato dos ficheiros *Wire* (fragmentos indicados na figura 2-8). Figura retirada de [18].

Quando a aplicação cliente pede um conteúdo ao servidor IIS, este procura no ficheiro MP4 o fragmento correspondente ao pedido do cliente (cada fragmento tem a sua posição temporal), e de seguida faz o levantamento desse fragmento e da informação a ele associada, enviando-o através da rede para o cliente. Este método melhora muito a eficiência do servidor IIS, pois elimina a necessidade de reescrever ou re-multiplexar informações de cabeçalho.

Numa transmissão utilizando Smooth Streaming estão tipicamente envolvidos os ficheiros:

Ficheiros MP4 com conteúdos vídeo/áudio

- *.ism
  - Contém vídeo e/ou áudio;
  - Um por cada *bit rate*.
- *.isma
  - Contém apenas áudio.

Ficheiro afecto ao servidor

*.ism

- Descreve as relações entre os conteúdos, bit rate e ficheiros armazenados;
- Apenas utilizado pelo servidor IIS Smooth Streaming.

Ficheiro afecto ao cliente

*.ismc

- Descreve as streams disponíveis para o cliente: os codecs utilizados, bit rates, resoluções do vídeo, entre outros;
- É o primeiro ficheiro a ser transmitido para o cliente.

No que diz respeito ao sistema de ficheiros, este protocolo adoptou formatos que permitiram uma melhoria significativa na optimização de recursos de armazenamento. Os vídeos, em vez de divididos fisicamente em milhares de segmentos, são divididos "virtualmente" em fragmentos (tipicamente um por GOP), e armazenados em apenas um ficheiro MP4. Esta característica implica duas grandes alterações no desenho do servidor e do cliente:

O servidor deve receber e interpretar pedidos URL para determinar a posição exacta de fragmentos dentro de um ficheiro MP4;

O cliente pode pedir segmentos de vídeo de uma forma que simplifica a implementação, por exemplo, por tempo em vez de identificador de sequência.

O início de uma sessão em Smooth Streaming segue os seguintes passos:

1. O cliente pede o ficheiro .ismc ao servidor;
2. Após a recepção do ficheiro, selecciona o conteúdo pretendido e prepara a reprodução do ficheiro tendo em conta os parâmetros de codificação (*codecs*, *bit rate*, resolução);
3. Envia um URL com o conteúdo seleccionado e parâmetros seleccionáveis (*bit rate* e posição), exemplo para ficheiro NB.ism, *bit rate* 40000, tempo 61027 ns:

```
http://vid.foo.com/NB.ism/QualityLevels(40000)/Fragments(video=61027)
```

4. Recebe pedido e consulta o .ism para saber a localização física do ficheiro .ismv/a pretendido;
5. Lê o ficheiro .ismv/a e procura o fragmento que contém a posição indicada pelo cliente;
6. Retira o fragmento e envia-o como um ficheiro independente que pode ser guardado noutros locais da rede para o caso de outro cliente enviar o mesmo pedido URL.

Relativamente à detecção da qualidade da rede, está a cargo da aplicação Silverlight do lado do cliente, o servidor não interfere neste passo. A aplicação cliente analisa os tempos de *download* dos fragmentos, estado de ocupação do *buffer*, entre outros indicadores, e decide quando deve pedir vídeos com maior ou menor qualidade. Se os vídeos com codificações diferentes estiverem bem sincronizados, a troca de *streams* será transparente para o utilizador.

2.1.5 - Adobe Dynamic Streaming

Em 2002, a Macromedia (actual Adobe Systems), lançou um novo servidor denominado Flash Communication Server MX, com o objectivo de fornecer um distribuidor de conteúdos multimédia para aplicações desenvolvidas sobre a tecnologia Flash.[19],[20]

Posteriormente, em Novembro de 2005, este servidor recebeu uma actualização não só ao nível das funcionalidades, mas também no seu nome, passando a ser denominado FMS (Flash Media Server). Nesta versão, foi introduzida uma funcionalidade que consistia na detecção automática da largura de banda disponível para cada utilizador. Esta era executada no início da sessão, ou seja, quando o utilizador requisitava um vídeo. Depois de executada, esta função retornava uma estimativa do valor da largura de banda disponível, possibilitando ao FMS a selecção do conteúdo que melhor se adaptava à qualidade da ligação do utilizador, para que lhe fosse fornecida uma boa QoE. Esta funcionalidade só era executada no início da sessão, pois era relativamente demorada e poderia causar interrupções na reprodução dos conteúdos. Apesar de já existir alguma adaptação às condições da rede, se ocorressem problemas durante a transmissão, a QoE do utilizador seria afectada sem que o sistema cliente/servidor tentasse algo para contrariar este facto.

Mais recentemente, em Janeiro de 2009, a Adobe Systems lançou a actual versão do seu servidor FMS, a 3.5. Nesta versão vem incluída uma funcionalidade que dá pelo nome de Dynamic Streaming. Ao contrário da versão 2.0, esta nova funcionalidade permite que o sistema cliente/servidor efectue a detecção de largura de banda disponível[21] e, se

necessário, possibilita a adaptação dos conteúdos enviados a qualquer momento durante a transmissão. A transição entre diferentes *streams* é realizada sem interrupções na reprodução, e de forma transparente para o utilizador. As aplicações do cliente (Adobe Flash Lite, Adobe Flash Player ou Adobe Air), comunicam com o servidor através do protocolo RTMP (Real Time Messaging Protocol).

O protocolo RTMP é um protocolo proprietário, criado pela MacroMedia, e foi desenvolvido com o objectivo de transmitir com alto desempenho sinais de áudio, vídeo e dados entre plataformas Adobe Flash, Adobe Flash Player e Adobe Air. Em Junho de 2009, a Adobe publicou um *draft* com as especificações deste protocolo[22], tornando-o acessível ao público. Dessa forma, possibilitou o desenvolvimento de novos produtos e tecnologias que permitem o envio a partir de um ou mais servidores, de conteúdos multimédia utilizando os formatos AMF, SWF, FLV, e F4V compatíveis com o Adobe Flash Player.

Funcionamento

Para perceber o funcionamento do serviço Adobe Dynamic Streaming foi configurado um servidor FMS, e implementado um cliente em Flash. Na implementação do reprodutor foi seguida uma solução já existente[23] e foi ainda incluída uma consola que apresenta as mensagens trocadas entre o cliente e o servidor.

Através dos dados recolhidos, foi representado o diagrama de estados representado na **Figura 2-10**. Foram também representadas todas as acções executadas em cada um dos estados, descritas seguidamente.

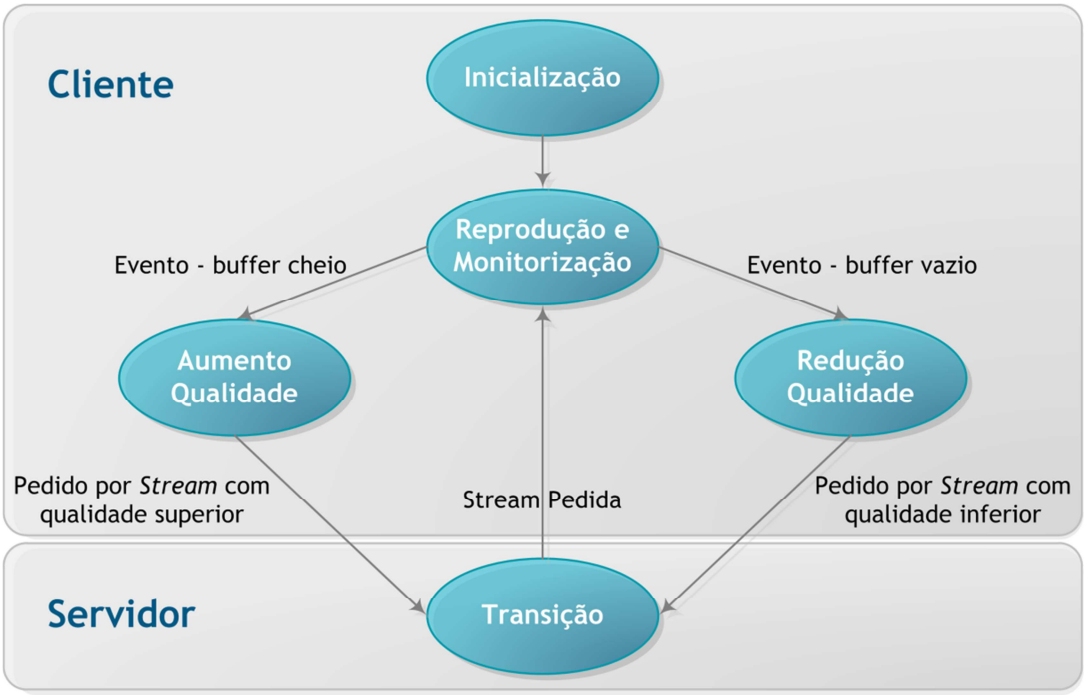

Figura 2-10 – Diagrama de estados para a detecção de largura de banda utilizando o protocolo Adobe Dynamic Streaming.

1. Inicialização

Figura 2-11 – Diagrama de estados da inicialização de uma sessão utilizando o protocolo Adobe Dynamic Streaming.

Neste estado, são geradas as listas que contêm as *streams* disponíveis para o cliente. Em cada entrada da lista, está indicado o *bit rate* de cada *stream*, para que, quando solicitado, o servidor saiba qual a *stream* a enviar para o cliente.

Quando a ligação ao FMS é estabelecida com sucesso, a função checkBandwidth(), implementada do lado do servidor, é invocada. Esta devolve para a função onBWDone(), o valor estimado da largura de banda disponível para o cliente, identificado pelo objecto criado anteriormente. O valor da largura de banda é passado para o cliente, e este escolhe, entre as entradas das listas de *streams*, a que melhor se adequa às condições da sua ligação.

2. Reprodução e Monitorização

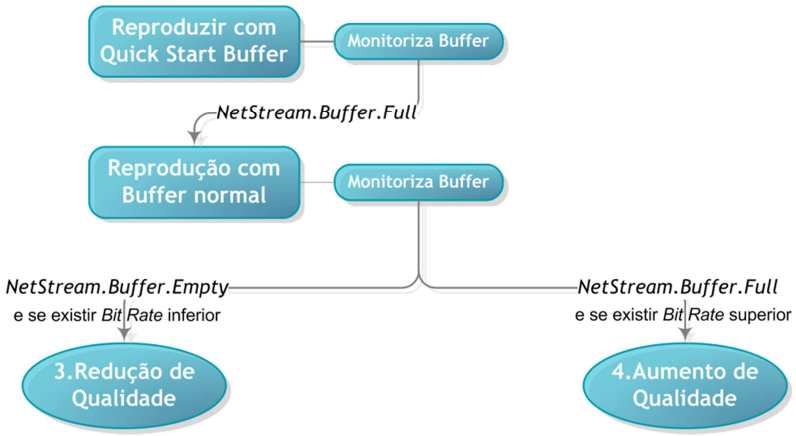

Figura 2-12 – Diagrama de estados da reprodução e monitorização de uma sessão utilizando o protocolo Adobe Dynamic Streaming.

Inicialmente, a recepção das *streams* é feita utilizando um *buffer* de rápida inicialização para que a reprodução seja iniciada com o menor atraso possível. Quando este enche, o seu tamanho é aumentado para que consiga receber mais dados, e assim possibilitar uma reprodução fluida.

Durante toda a reprodução dos conteúdos, o *buffer* é monitorizado constantemente pela aplicação cliente. Caso seja detectado um *NetStream.Buffer.Full*, ou seja, se a taxa de recepção for superior à de reprodução, é iniciado o processo de aumento de qualidade da *stream* enviada pelo servidor. Pelo contrário, se for detectado um evento *NetStream.Buffer.Empty*, isto é, não existirem dados no *buffer* e como consequência ocorrerem interrupções na reprodução, é iniciado o processo de redução de qualidade.

3. Aumento de Qualidade

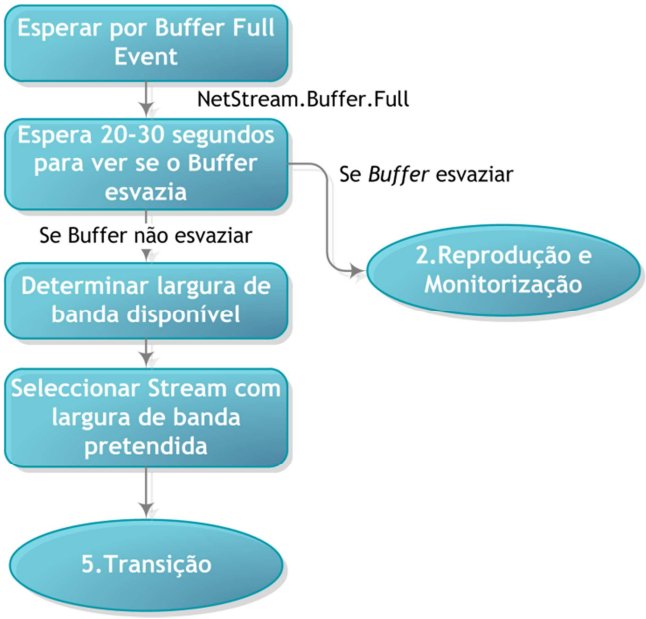

Figura 2-13 – Diagrama de estados do processo de aumento de qualidade de conteúdos utilizando o protocolo Adobe Dynamic Streaming.

Durante a reprodução, se forem detectados eventos *NetStream.Buffer.Full*, significa que o *buffer* enche. Idealmente, a taxa de recepção deve ser igual à taxa de reprodução, ou seja, o *buffer* nem enche nem esvazia, pois a taxa a que os dados são reproduzidos, isto é, retirados do *buffer*, é a mesma a que novos dados são recebidos por ele. Assim, quando esta situação ocorre repetidamente, significa que podemos aumentar o *bit rate*, e consequentemente a QoE do utilizador.

Durante o processo de aumento de qualidade, a aplicação verifica se realmente é possível aumentá-la, isto é, se o *buffer* realmente tem capacidade de receber a maiores taxas, ou se foi uma situação pontual. Caso seja detectada a possibilidade de aumento, é determinada a largura de banda utilizando o método checkBandwidth() do FMS, e é escolhida uma *stream* de acordo com o valor retornado por essa função. Depois de escolhida a *stream*, é enviado o pedido de transição para o servidor.

4. Redução de Qualidade

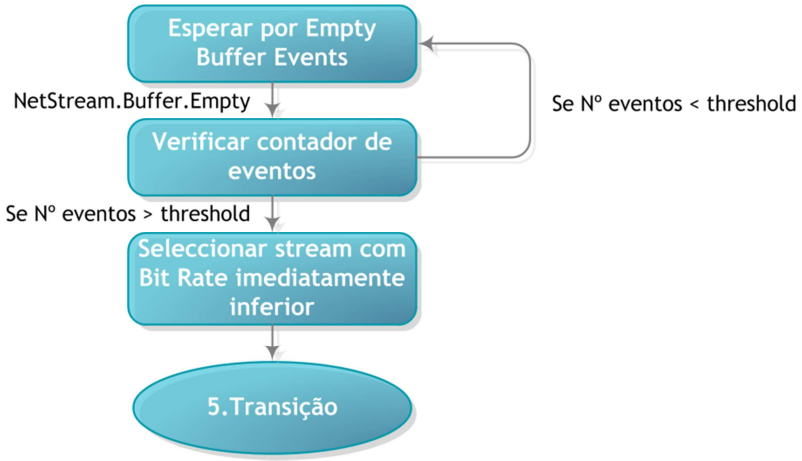

Figura 2-14 – Diagrama de estados do processo de redução de qualidade de conteúdos utilizando o protocolo Adobe Dynamic Streaming.

Caso sejam detectadas situações em que o *buffer* esvazia, ou seja, a aplicação deixa de ter conteúdos para reproduzir (o que origina interrupções na reprodução), é iniciado o processo de redução de qualidade. Neste processo, como acontece com o aumento, é verificado se realmente existe a necessidade de reduzir a qualidade. Caso ocorra um determinado número de eventos vazios, a aplicação do cliente escolhe uma *stream* com qualidade/*bit rate* inferior, para que o *buffer* consiga receber sem ocorrerem falhas. Depois de escolhida a *stream*, é enviado o pedido de transição para o servidor.

5. Transição

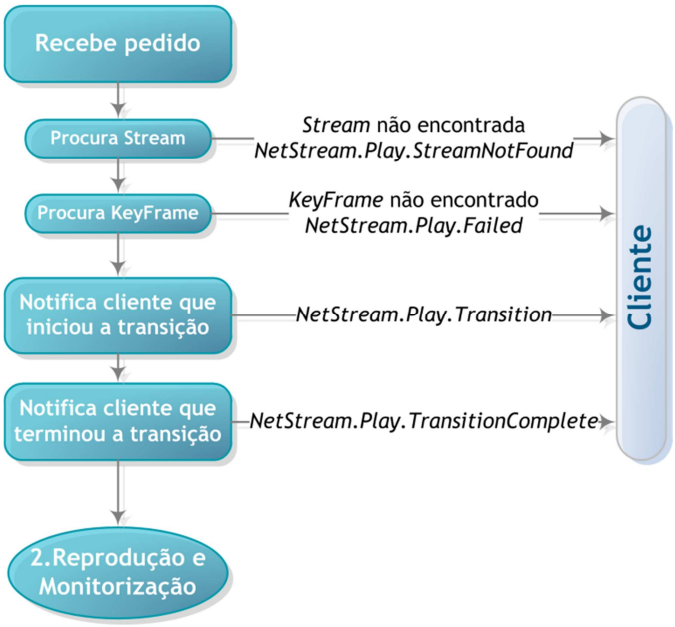

Figura 2-15 – Diagrama de estados da transição entre conteúdos de diferentes qualidades utilizando o protocolo Adobe Dynamic Streaming.

Após a recepção do pedido de transição enviado pelo cliente, o servidor verifica se o pedido é válido, ou seja, se a *stream* existe. Caso não exista, envia uma mensagem de erro conhecida para o cliente. Caso exista, é procurado o próximo *keyframe* dentro do vídeo, sendo que a transição só é realizada se puder ser feita com sincronismo, ou seja, caso existam *keyframes* em ambas as *streams* no mesmo ponto. Desta forma, a transição é realizada de forma transparente para o utilizador, ou seja, sem interrupções na reprodução.

Quando é detectada a proximidade do *keyframe*, e consequentemente da transição, o servidor envia uma notificação para o cliente, *NetStream.Play.Transition*, indicando a troca de *streams*. Da mesma forma, quando é iniciada a transmissão da nova *stream*, é enviada outra notificação, *NetStream.Play.TransitionComplete*.

2.1.6 - Análise Comparativa

Tabela 2-1 — Soluções de *streaming*, dados recolhidos de [11],[12],[13],[16],[17],[18],[23],[24].

	RTSP/RTP	Adobe Dynamic Streaming	HTTP-Live Streaming	Smooth Streaming
Tipo de Streaming	Tradicional	Streaming Adaptativo	Streaming Adaptativo	Streaming Adaptativo
Protocolo de Streaming	RTP	RTMP	HTTP	HTTP
Protocolo de Transporte	TCP/UDP	TCP	TCP	TCP
Plataformas Suportadas	Player RTSP	Flash Player 10, Adobe Air	iPhone OS 3.0, QuickTime X-MacOS	Silverlight, iPhone OS 3.0
Emissor de mensagens	Cliente/ Servidor	Cliente/Servidor	Cliente	Cliente
Transmissões em directo (atraso)	Codificação (< 2s)	> 6s	> 30s	> 2s
Determinação da largura de banda	Não normalizado	Cliente/Servidor	Cliente	Cliente
Tipo de ligação	Stateful	Stateful	Stateless	Stateless
Compatibilidade com a rede	Reduzida (bloqueado por alguns firewalls)	Reduzida (bloqueado por alguns firewalls)	Elevada	Elevada
Escalabilidade	Reduzida (uma sessão por utilizador)	Reduzida (uma ligação FMS por utilizador)	Elevada	Elevada
Formato ficheiros	-	MPEG 4 – part 12, FLV	MPEG 2 TS	MPEG 4 – part 12
Tipo de armazenamento	Contínuo	Contínuo (1 por bitrate)	Fragmentado (N por bitrate)	Contínuo (1 por bitrate)
Espaço em disco necessário	Normal	Reduzido (ficheiros FLV)	Elevado (N segmentos por bit rate)	Normal para VoD, elevado para Live
Codecs de Vídeo	Limitado pelo cliente	H.264 Baseline, Main, e High; VP6	H.264 Baseline Level 3.0	VC-1, H.264 (MPEG-4 AVC)
Codecs de Áudio	Limitado pelo cliente	AAC, MP3	MP3, HE-AAC, AAC-LC	WMA, AAC

Através da análise da **Tabela 2-1**, é importante realçar:

RTSP/RTP

Uma das características diferenciadoras deste protocolo é a independência do protocolo de transporte. Enquanto os restantes protocolos são associados apenas a TCP, os pacotes RTP podem ser transportados sobre UDP, eliminando possíveis atrasos provenientes das retransmissões TCP, mas com a desvantagem de poder ocorrer perda de informação. Também por causa do funcionamento sobre UDP, as mensagens enviadas por este protocolo são bloqueadas por alguns *firewalls*, principalmente em ambiente empresarial onde o controlo é mais restrictivo.

Outra característica importante é existência de comunicação entre o cliente e o servidor, ou seja, existem respostas de parte a parte. Desta característica advém a baixa escalabilidade deste protocolo que, por ser *stateful*, requer uma comunicação dedicada com cada utilizador.

Finalmente, uma vantagem relevante deste protocolo é a segurança contra cópia dos conteúdos transmitidos. Uma vez que não existe armazenamento do lado do cliente, e principalmente porque não suporta *caches* ou *proxies* HTTP, o acesso aos conteúdos transmitidos por parte de utilizadores sem permissões para tal é dificultado.

Adobe Dynamic Streaming

Uma das principais vantagens deste tipo de transmissão é a compatibilidade do seu *player*. Em Março de 2010, as estimativas apontavam que 97% dos computadores pessoais nos principais mercados tinham instalado a última versão do Adobe Flash Player (i.e. Adobe Flash Player 10)[25], que suporta Dynamic Streaming. Por isso, a compatibilidade deste protocolo ao nível da reprodução é superior aos seus concorrentes.

Assim como no protocolo RTSP/RTP, também no Adobe Dynamic Streaming existe comunicação entre cliente/servidor. E também como em RTSP/RTP essa característica causa problemas de escalabilidade para este protocolo, uma vez que, para cada utilizador, é necessária uma ligação ao servidor e, consequentemente, é necessária maior capacidade de processamento e memória.

Uma grande vantagem deste protocolo relativamente aos outros de *streaming* adaptativo (i.e. HTTP-Live Streaming e Smooth Streaming) é o facto de tanto o servidor como o cliente terem acesso à estimativa da largura de banda disponível para o cliente. Desta forma é possível que tanto o servidor como o cliente adaptem os conteúdos a transmitir, ao contrário dos outros protocolos onde essa função está apenas disponível/afecta ao cliente.

HTTP-Live Streaming

Este protocolo, ao contrário dos anteriores, é transportado sobre HTTP, ou seja, a sua compatibilidade com a rede é bastante superior, pois a maioria dos *routers* e elementos de rede foram optimizados para o transporte de dados sobre HTTP. Já a compatibilidade com os reprodutores de conteúdos é reduzida, visto que apenas suporta duas plataformas, ambas proprietárias da Apple (iPhone, ou MAC OS).

A sua escalabilidade é elevada, uma vez que é um protocolo *stateless*, ou seja, apenas interpreta os pedidos do cliente e responde aos mesmos, sem qualquer tipo de relação com

os anteriores. Como neste protocolo o servidor não conhece os clientes, não existe comunicação entre eles.

O facto de funcionar sobre HTTP implica outra funcionalidade para este protocolo, a possibilidade de armazenamento dos seus conteúdos em *caches* ou *proxies* distribuídos pela rede. Desta forma a carga sobre o servidor é diminuída, pois se ocorrerem dois pedidos pelos mesmos conteúdos, o servidor não necessita de responder ao segundo, visto que os conteúdos já existem na rede. Embora esta seja uma grande vantagem deste protocolo, também se apresenta como uma desvantagem para a segurança dos conteúdos, visto que existem cópias dos conteúdos espalhadas pela rede em caches HTTP.

Duas desvantagens deste protocolo são o atraso induzido em transmissões em directo[2], e a elevada capacidade de armazenamento necessária, uma vez que para cada *bit rate* existem muitos ficheiros que formam um conteúdo/programa.

Uma grande vantagem é o facto de os conteúdos poderem ser transmitidos por um servidor *web* comum, sendo que se baseia no simples envio de ficheiros para o cliente.

Smooth Streaming

Este protocolo é bastante semelhante ao HTTP-Live Streaming, sendo que as suas vantagens e desvantagens relativas ao protocolo de transporte, escalabilidade, compatibilidade são idênticas.

A principal vantagem deste protocolo sobre o HTTP-Live Streaming é o seu sistema de ficheiros que proporciona grandes poupanças no que diz respeito ao armazenamento, visto que, ao contrário do HTTP-Live Streaming, apenas existe um ficheiro para cada *bit rate* funcionando a pesquisa de segmentos pedidos pelo cliente através de apontadores para dentro desse ficheiro.

Outro factor que diferencia este protocolo do HTTP-Live Streaming é o seu reduzido atraso para transmissões em directo, que corresponde apenas ao tempo necessário para codificar os conteúdos.

Também o tipo de servidor *web* é diferente, uma vez que neste protocolo, é necessário que o servidor consiga interpretar o sistema de ficheiros, sabendo como procurar fragmentos dentro do ficheiro principal.

2.2 - Necessidade da Solução

Após o levantamento do estado da arte, verificou-se que todas as soluções e/ou serviços de *streaming* que possuem adaptação de *bit rate* apresentam algumas limitações significativas no contexto desta dissertação. Uma das principais é a dependência que os sistemas apresentam relativamente a aplicações instaladas nos terminais dos utilizadores.

Outra das desvantagens dos serviços encontrados reside no facto de necessitarem que os conteúdos sejam previamente codificados com diferentes *bit rates*, mesmo que os segmentos

[2] Assumindo que cada segmento tem a duração de 10 segundos como recomendado, os últimos 20 segundos da actual transmissão estarão inacessíveis ao utilizador. Antes de ser iniciada a reprodução pelo menos dois ficheiros devem ser obtidos pelo cliente para uma reprodução fluida.

resultantes não sejam utilizados posteriormente. Esta característica diminui a eficiência do sistema, uma vez que gasta recursos na codificação, que poderão ser inúteis. Ainda outra consequência deste tipo de abordagem é a redução da orientação ao cliente, ou seja, não é entregue ao utilizador o que ele realmente necessita.

Relativamente à integração em ambientes com muitos utilizadores, as soluções encontradas também não são as ideais, pois consomem muitos recursos no armazenamento dos conteúdos. Como a solução a desenvolver no decorrer desta dissertação tem como objectivo a inserção nesse mesmo ambiente é necessária uma solução mais eficiente. Essa solução passa pela aplicação do protocolo RTSP sobre UDP como será explicado no próximo capítulo.

Capítulo 3

Tecnologias e Componentes

Neste capítulo é justificada a escolha dos protocolos utilizados no módulo de adaptação e são ainda apresentadas as tecnologias que o suportam. São também descritos alguns conceitos importantes para as tomadas de decisão na implementação da solução desenvolvida.

3.1 - WSS (Windless Streaming Server)

O WSS[26] é uma *gateway* para *streaming* de conteúdos, em tempo real ou a pedido, acessíveis via Internet ou redes móveis. Para além das funcionalidades básicas de RTSP *proxy*, o WSS é capaz de fazer a adaptação (*transcoding*, *variable bit rate*, *resizing*) e manipulação (*text overlay*, *animated gifs*, *logo insertion*, *pre/mid/end-roll clips*) de *streams* em tempo real.

O valor do WSS, no encaminhamento dos *streams* de áudio e vídeo de conteúdos multimédia de múltiplas fontes para os terminais móveis, assenta nos seguintes princípios:

- Maximizar as capacidades dos terminais, assegurando sempre a melhor qualidade do vídeo, actuando na adaptação do *bit rate* e fazendo *transcoding* em tempo real de qualquer formato (*codec*, *bit rate*, *frame rate*, etc);
- Facilitar a gestão e alojamento de conteúdos, necessitando apenas de uma cópia de determinado conteúdo (ficheiro ou *live stream*);
- Maximizar a atenção dos utilizadores durante a publicidade, assegurando assim uma maior retenção do produto através de recurso a *banners*, *gifs* animados e inserção de anúncios antes, durante e depois dos clips;
- Personalizar o serviço recorrendo à inserção de logos, *text overlay* e permitindo a comutação entre *streams* de forma transparente.

O WSS controla as sessões de *streaming*, estabelecendo a ligação entre um utilizador de um terminal móvel e um fornecedor de conteúdos, numa rede local ou na Internet. É possível fornecer conteúdos de um canal de televisão em directo, acesso a um vídeo blogue, etc.

É possível realizar a inserção de um anúncio antes de fornecer o conteúdo que o cliente solicitou, e também realizar uma separação total entre a rede do cliente e a dos servidores de conteúdos. Durante uma sessão de *streaming*, os clientes e os servidores de conteúdos só vêem o WSS, desconhecendo ambos a existência um do outro.

3.2 - Protocolo de streaming utilizado

Como referido anteriormente, durante uma transmissão sobre qualquer tipo de ligação, dificilmente a qualidade da rede se mantém constante. Este factor é ainda mais evidente em redes móveis, como as redes 3G actualmente em utilização. Existem muitas causas para este comportamento, como a alteração entre células a que o cliente está ligado, a entrada num edifício ou transporte público por exemplo. Todos estes factores contribuem para que a qualidade da rede se degrade, causando erros na transmissão visíveis na reprodução dos conteúdos. Por esse motivo revela-se necessária a existência de mecanismos de adaptação dos conteúdos transmitidos de modo a que estes se adaptem às reais condições da rede.

Alguns dos protocolos aqui apresentados já possuem os seus métodos de determinação de largura de banda implementados. Por exemplo, os protocolos HTTP-Live Streaming e Smooth Streaming fazem ambos a detecção do lado do cliente, sem que o servidor tenha qualquer conhecimento ou função neste processo. Nestes dois casos, as aplicações do lado do cliente, depois de realizaram a detecção, enviam um pedido ao servidor por um segmento de vídeo já com qualidade ajustada à largura de banda disponível.

Também no protocolo Dynamic Streaming, nomeadamente através da utilização do Flash Media Server, já existem métodos implementados que permitem o cálculo da largura de banda através da monitorização do *buffer* do terminal receptor dos conteúdos.

Como um dos principais objectivos do módulo a desenvolver é manter toda a implementação do lado do servidor (justificado mais à frente neste documento), os anteriores protocolos ficam em desvantagem.

Deve ainda ser feita uma análise ao tipo de adaptação que os protocolos apresentados permitem. Em todos os anteriores protocolos, é enviada para o terminal receptor uma lista com os conteúdos que o utilizador pode escolher. Depois do utilizador seleccionar o conteúdo que pretende visualizar, a aplicação cliente consulta outra lista que contém diferentes ficheiros relativos ao conteúdo seleccionado, cada um deles com diferentes parâmetros de codificação. Isto significa que, para a adaptação ser possível, antes de disponibilizar a lista de conteúdos o servidor deve possuir ficheiros pré-codificados com diferentes débitos binários. Este requisito implica duas desvantagens imediatas: não é uma abordagem *client-oriented*, pois não é a transmissão que se adapta ao cliente, mas sim o cliente que tem de escolher de uma lista pré-definida; para cada conteúdo existem múltiplos ficheiros, o que aumenta o armazenamento necessário e o consumo de recursos.

Com a implementação de uma nova solução de adaptação sobre RTSP/RTP utilizando o WSS, é possível eliminar as anteriores desvantagens codificando os conteúdos em tempo-real, e de acordo com a necessidade de cada cliente.

Assim sendo, a solução a desenvolver deverá focar o protocolo RTSP/RTP.

3.3 - Protocolo de transporte - TCP vs UDP

Na descrição de RTSP/RTP efectuada no levantamento do estado da arte, foi referido que ao contrário dos restantes protocolos, RTSP/RTP é independente do protocolo de transporte, ou seja, as suas mensagens e pacotes tanto podem ser transportadas sobre UDP ou TCP. Assim, é importante referir as principais diferenças entre eles, uma vez que a sua utilização influencia muito a transmissão, e consequentemente a forma como a solução foi pensada.

Os protocolos TCP e UDP pertencem à camada quatro do modelo OSI[27] (camada de transporte) e, em conjunto com a porta e IP da máquina, definem como uma determinada informação é transmitida na rede.

3.3.1 - TCP (Transmission Control Protocol)

Este é um dos principais protocolos em redes TCP/IP. Enquanto que o protocolo IP trata apenas de pacotes, TCP possibilita que dois elementos de rede estabeleçam uma ligação, e troquem fluxos de dados. Uma das propriedades do TCP que o torna tão popular é a garantia de entrega dos dados sem erros, e na mesma ordem com que foram enviados[28].

TCP é um protocolo *connection-oriented*, ou seja, orientado à conexão e é responsável pela comunicação fiável entre dois intervenientes. Por ser orientado à conexão, os fluxos de dados transmitidos representam uma sequência anónima de *bytes*, ou seja, não existem fronteiras identificáveis nos pacotes de dados. Assim, o receptor não tem forma de saber como os dados foram transmitidos, apenas num pacote grande ou em muitos pequenos. Pode ainda acontecer que o emissor envie um pacote e este seja dividido durante a transmissão. Isto significa que o receptor deve estar pronto para receber e descodificar qualquer tipo de fluxo de dados, sendo que a única garantia que possui é que estes chegarão completos e sem erros.

Para cada pacote recebido, é enviada uma resposta para o emissor designada "ACK", informando-o se a recepção foi bem concluída ou não. Assim, por cada pacote de dados transmitido, pelo menos mais um é introduzido na rede, isto porque, caso existam erros na transmissão, serão enviados mais pacotes, como veremos de seguida.

Para que a fiabilidade deste protocolo fosse possível, foi necessário implementar um mecanismo reactivo à ocorrência de erros, uma vez que não é possível preveni-los. O mecanismo implementado baseia-se na retransmissão de pacotes quando estes são perdidos na rede ou chegam ao seu destino com erros. A **Figura 3-1** representa a troca de mensagens durante uma transmissão de pacotes codificados sobre TCP.

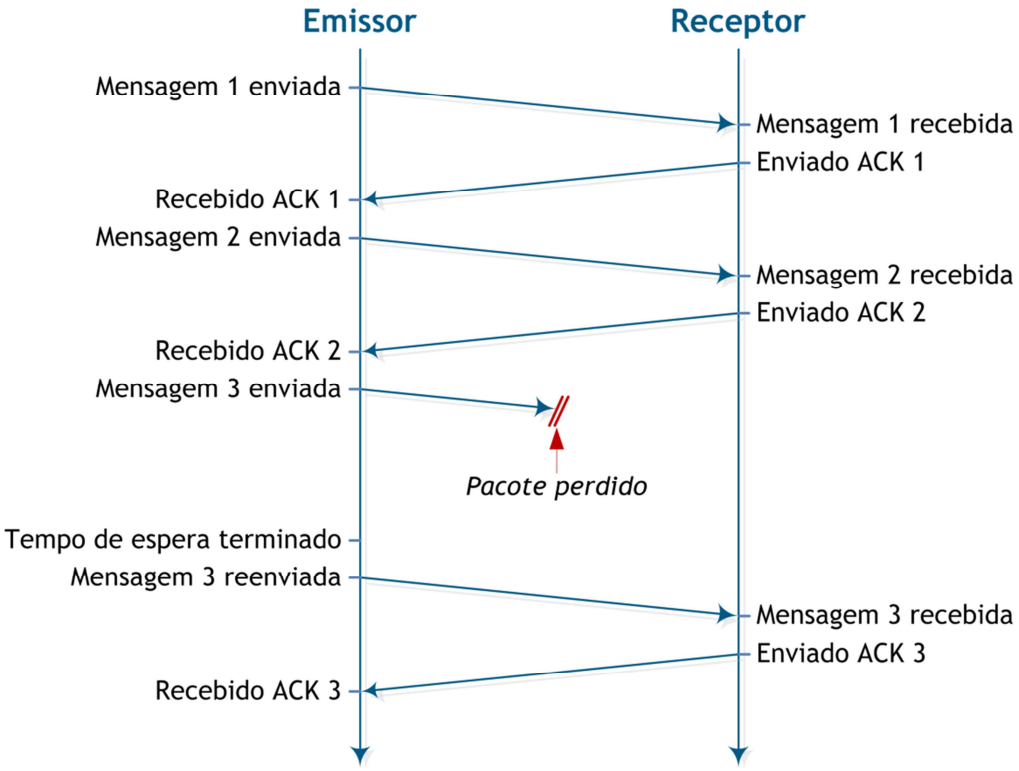

Figura 3-1 – Transmissão sobre TCP.

Devido às verificações de conformidade necessárias para cada pacote recebido, às retransmissões causadas por cada erro, à necessidade de confirmação da recepção e consequente aumento da quantidade de informação presente na rede, a velocidade da transmissão do protocolo TCP é diminuída.

3.3.2 - UDP (User Datagram Protocol)

Ao contrário de TCP, este protocolo é do tipo *connectionless*, não orientado à conexão e como TCP, funciona sobre redes IP. A principal diferença relativamente a TCP é o facto de não possuir mecanismos de recuperação de erros, em vez disso possibilita uma troca directa de datagramas através de uma rede IP. Este protocolo é utilizado principalmente para fazer *broadcast* de mensagens numa rede[29].

Por ser um protocolo não orientado à conexão, os seus pacotes/datagramas podem ser enviados a qualquer momento, sem nenhum aviso prévio nem negociação. O datagrama é simplesmente enviado independentemente do estado da recepção do cliente, ou seja, mesmo que este não tenha recebido com êxito os anteriores datagramas, o emissor não altera a transmissão. Consequentemente, os pacotes perdidos não serão recuperados ao contrário do protocolo TCP. Devido a este facto, UDP é um protocolo não fiável.

Para além de existir a possibilidade da transmissão dos datagramas não ser concluída com êxito, os mesmos podem ainda chegar fora de ordem. Ou seja, é possível receber primeiro um datagrama que cronologicamente é apresentado depois de outro ainda não recebido. É

ainda possível receber pacotes duplicados. Assim sendo, as aplicações receptoras devem estar prontas para detectar e contornar este tipo de anomalias.

No entanto, esta característica de UDP implica que o cliente seja capaz de identificar correctamente todos os pacotes, ou seja, estes devem ter as suas fronteiras bem delineadas e identificadas, facilitando assim a tarefa de descodificação do cliente.

Devido ao seu funcionamento *stateless*, ao contrário de TCP, o receptor não envia confirmação de recepção ou avisos de erro para o emissor, consumindo menos recursos de rede, deixando assim mais capacidade para os dados propriamente ditos. Esta propriedade aumenta também a capacidade de um determinado emissor relativamente ao número de receptores a que se liga, uma vez que não necessita de criar uma ligação dedicada para cada um deles, podendo transmitir os dados por broadcast.

Como o emissor não necessita de esperar pela confirmação de recepção por parte do receptor, nem precisa retransmitir pacotes passados, as velocidades de transmissão são mais elevadas relativamente ao protocolo TCP.

3.3.3 - Decisão

Quanto à aplicabilidade de UDP ou TCP ao *streaming* de conteúdos multimédia, face às propriedades de cada um é possível concluir que com a utilização de TCP, ocorrerão atrasos na transmissão, provocando assincronismo e eventualmente interrupções na reprodução. No entanto, com a utilização de UDP, serão perdidos pacotes, implicando degradação visível na reprodução.

Quando aplicada a cenários multi-utilizador nos quais centenas ou milhares de utilizadores estarão ligados ao mesmo servidor, a utilização do protocolo TCP origina maiores cargas introduzidas na rede, transmitindo pelo menos duas mensagens ou pacotes para cada transmissão de dados.

Após a análise aos dois protocolos, é possível concluir com alguma certeza que o protocolo UDP é o mais adequado ao cenário pretendido. Uma vez que o objectivo desta dissertação é melhorar a QoE do utilizador final em cenários onde a ligação tem menor qualidade. Neste caso, a probabilidade de ocorrência de erros é bastante elevada, originando sobrecargas na rede, e principalmente atrasos consideráveis na reprodução de conteúdos caso fosse utilizado o protocolo TCP. A utilização de UDP reduz também a necessidade de processamento para o servidor, principalmente em casos de transmissões em directo, onde para todos os clientes são enviados os mesmos dados.

Apesar de à partida o protocolo UDP ser recomendado na implementação de sistemas de *streaming* com RTSP/RTP, foram encontradas implementações com TCP, em que os cálculos relativos à largura de banda são efectuados utilizando o campo RTT (Round-Trip delay Time) que representa o tempo desde que um pacote foi enviado até que a resposta relativa a esse pacote foi recebida. Sabendo este valor, e tendo a garantia de entrega de todos os pacotes

transmitidos, a detecção da qualidade de transmissão poderá ser efectuada com maior certeza relativamente ao cenário com UDP.

Um factor importante que contribuiu bastante para a escolha do protocolo UDP foi o facto da implementação existente no WSS seguir as recomendações definidas no RFC do protocolo RTSP, ou seja, utiliza o protocolo UDP. Assim, o desenvolvimento e realização de testes ao módulo desenvolvido são possíveis utilizando cenários reais em ambiente controlado.

3.4 - Aplicação no cliente

Uma das possíveis abordagens para a solução pretendida seria desenvolver uma aplicação a instalar no terminal do cliente. Dessa forma seria possível monitorizar a recepção dos dados recorrendo ao *buffer* do cliente sendo possível obter uma medição mais precisa do valor da largura de banda utilizada. Seria ainda possível definir o intervalo de tempo em que haveria trocas de mensagens, possibilitando adaptações em tempo-real.

Resumidamente seria possível definir e especificar toda a comunicação, não existindo dependência de outros componentes, obtendo assim maior controlo sobre o sistema.

Este foi o método adoptado pelos restantes protocolos ou serviços existentes, nomeadamente:

- HTTP-Live Streaming com QuickTime Player;
- Smooth Streaming com Silverlight;
- Adobe Dynamic Streaming com Adobe Flash Player.

Em ambientes móveis, uma das principais desvantagens dos serviços mencionados prende-se com o facto de existir um número muito elevado de diferentes modelos de terminais. Sendo que um dos principais objectivos desta dissertação é desenvolver um módulo com elevada compatibilidade, seria necessário implementar diferentes aplicações para cada tipo de plataforma (Android, iPhone OS, Symbian OS, computadores portáteis, entre outros). Assim, a abordagem escolhida foi manter toda a implementação do lado do servidor, tornando-a independente do tipo de terminal. Tomando esta opção, evitamos ainda que o cliente seja forçado a instalar uma aplicação específica para utilizar o serviço.

3.5 - Estimação da largura de banda utilizada

É necessário realçar que esta é apenas uma estimativa/referência e não um valor exacto, pois apenas temos em consideração o número de *bytes* enviados pelo nosso servidor, ignorando todos os restantes conteúdos que se encontram em transmissão na rede. Também é feita uma aproximação do tamanho de cada pacote como é perceptível nos cálculos. Para que fosse possível obter um valor preciso, era necessário saber dados a que apenas o cliente tem acesso, e o seu envio não está padronizado, ou seja, seria necessário possuir uma aplicação do lado do cliente.

Durante uma transmissão/sessão RTSP, o cliente envia periodicamente os relatórios de recepção RTCP. O servidor, neste caso o WSS, retira desses relatórios o valor do *interarrival jitter* enviando-os para o módulo de adaptação para comparação com as anteriores leituras. Caso o módulo detecte que pode existir algum problema com a rede, ou seja, caso o valor do *jitter* aumente significativamente, o servidor dá início à adaptação do *bit rate*.

Este processo inicia-se com a gravação dos valores das seguintes variáveis no módulo de adaptação:

- Número de sequência do último pacote RTP recebido (relatório RTCP);
- *Timestamp* da recepção do último relatório RTCP;
- Contador do número de *bytes* enviados para o cliente (componente do WSS).

De seguida, o módulo de detecção espera até que seja recebido um novo relatório de recepção. Quando este é recebido, são registados os mesmos valores que anteriormente, mais o número de pacotes perdidos.

De seguida o módulo de detecção efectua os seguintes cálculos:

```
[intervalo entre relatórios RTCP] = [timestamp 2ºrelatório] – [timestamp 1ºrelatório]
[bytes enviados] = [valor final do contador] – [valor inicial do contador]
[pacotes enviados] = [nº sequência RTP final] – [nº sequência RTP inicial]
[tamanho médio dos pacotes] = [bytes enviados] / [pacotes enviados]
[bytes recebidos] = [bytes enviados] – [pacotes perdidos] × [tamanho médio dos pacotes]
[largura de banda] = [bytes recebidos] × 8 / [intervalo entre relatórios RTCP]
```

Através desta estimativa, é possível fazer a adaptação do *bit rate* do vídeo transmitido tendo algum valor como referência diminuindo assim a ocorrência de reduções exageradas, e instabilidade na recuperação.

3.6 - *Interarrival Jitter*

O *interarrival jitter* é uma estimativa da variância estatística dos tempos de chegada entre dois pacotes de dados RTP seguidos. É medido em unidades de tempo com formato *timestamp* e é expresso como um número inteiro sem sinal ([13], secção 6.4.1).

O *interarrival jitter* J é definido como o desvio médio (valor absoluto suavizado) da diferença D do espaçamento de pacotes no receptor comparado com a do emissor para o mesmo par de pacotes RTP. Como mostrado na **Equação 3-1**, é o equivalente à diferença da duração relativa de transmissão para o par de pacotes; a duração relativa de transmissão é a diferença entre o *timestamp* contido no pacote RTP e o valor do relógio do sistema no momento da recepção, medidos nas mesmas unidades([13], secção 6.4.4).

A **Figura 3-2** representa as variações do *jitter* durante o envio de pacotes RTP. Estão ainda representados os momentos considerados na **Equação 3-1**.

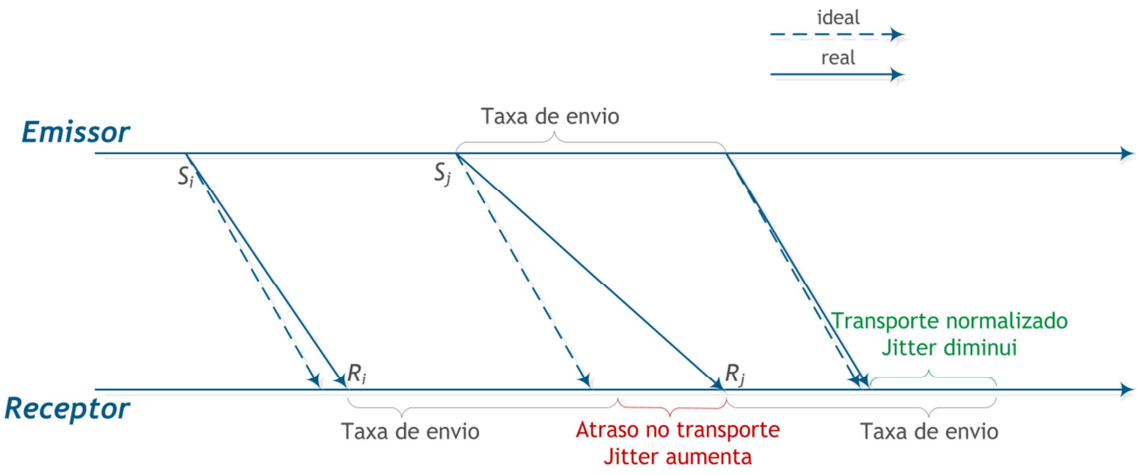

Figura 3-2 – Variações do *jitter*.

Se S_i for o *timestamp* do pacote RTP i, e R_i o tempo da recepção nas mesmas unidades, então para dois pacotes i e j, D pode ser expresso como

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \qquad \textbf{Equação 3-1}$$

O valor do *interarrival jitter* deve ser calculado continuamente, à medida que cada pacote de dados i é recebido do emissor, utilizando a anterior diferença D para esse pacote e o anterior, $i-1$ por ordem de chegada (não necessariamente de sequência), de acordo com a fórmula

$$J(i) = J(i-1) + \frac{(|D(i-1,i)| - J(i-1))}{16} \qquad \textbf{Equação 3-2}$$

Sempre que um relatório de receptor for emitido, o valor actual de J é amostrado e incluído no mesmo.

O *interarrival jiter* representa uma medida a curto-prazo do congestionamento da rede. Enquanto o número de pacotes perdidos indica congestionamento persistente, o *jitter* indica congestionamento transitório, ou seja, é possível detectar o momento em que a rede se degrada. Assim, através da monitorização das variações do *jitter*, é possível detectar congestionamento da rede antes que este implique perdas de pacotes.

Por ser apenas uma amostra do valor do *jitter* no momento de envio do relatório RTCP, o *interarrival jitter* recebido pelo emissor não deve ser considerado quantitativamente. Pelo contrário, destina-se a possibilitar comparações entre vários relatórios.

Pelas razões anteriormente descritas, o *interarrival jitter* revela-se importante para esta dissertação uma vez que indica a degradação momentanea da rede, possibilitando ainda o desenvolvimento de um algoritmo não apenas reactivo a erros, mas sim um pouco preventivo relativamente a perdas de pacotes.

3.7 - Linguagem Lua

Lua[30],[31] é uma linguagem de programação criada em 1993 no Tecgraf, um laboratório de pesquisa e desenvolvimento da PUC-Rio. É uma linguagem de *script* dinâmica, semelhante a outras linguagens de *script* como Python, Ruby e PHP.

Lua destaca-se pela sua simplicidade, portabilidade, rapidez e pela facilidade com que podemos embutir um interpretador Lua numa aplicação C. Outra característica importante de Lua é o seu reduzido tamanho. O núcleo da linguagem somado às suas bibliotecas padrão ocupa menos de 200 Kbytes. Lua é uma linguagem de uso geral, que pode ser utilizada tanto para escrever pequenos *scripts*, com poucas linhas de código, como sistemas complexos, com dezenas de milhares de linhas de código.

Actualmente a linguagem é utilizada principalmente em jogos, como World of Warcraft, The Sims e Sim City. Contudo, Lua não é uma linguagem específica para jogos, sendo utilizada também em muitas outras aplicações, como o *software* Adobe Photoshop Lightroom e o *middleware* Ginga do Sistema Brasileiro de TV Digital.

A escolha desta linguagem de programação para a implementação dos algoritmos do módulo desenvolvido deveu-se principalmente ao facto da mesma, juntamente com a linguagem C possibilitar que, após a alteração dos *scripts* implementados, seja possível executá-los sem ser necessário recompilar o código de todo o módulo. Assim, podemos introduzir actualizações no módulo sem ter de interromper o seu funcionamento.

Outra propriedade com especial interesse é a sua fácil integração com a linguagem C que, como anteriormente mencionado, foi a linguagem adoptada no desenvolvimento do WSS.

Capítulo 4

Arquitectura e Implementação

Este capítulo apresenta a solução proposta para a implementação de um módulo de adaptação com a descrição da arquitectura utilizada, e as funcionalidades e objectivos de cada componente. Em seguida, são descritas com detalhe todas as implementações efectuadas. Posteriormente, é apresentada uma secção onde são descritos todos os testes realizados aos componentes desenvolvidos, juntamente com os resultados da aplicação da solução e respectiva análise. Finalmente é apresentada a validação da solução desenvolvida, que inclui uma avaliação subjectiva por parte de diferentes participantes externos.

4.1 - Concepção

Em todo o processo de concepção houve sempre um objectivo em mente, manter a solução modular. A razão para esta opção prende-se com o facto da solução possuir algumas dependências de serviços e sistemas externos, como por exemplo, o reprodutor multimédia do cliente. Ainda mais evidente, é a dependência do módulo face à qualidade da ligação entre os utilizadores e o WSS. Sendo que a degradação da rede depende directamente de factores como o número de utilizadores ligados, distância aos utilizadores, até mesmo condições ambientais, deve ser possível alterar facilmente os parâmetros do módulo desenvolvido.

No entanto, as constantes actualizações necessárias não foram a única razão para a implementação modular escolhida, também o facto de existirem problemas bastante diferentes associados à solução: a obtenção da estimativa da largura de banda de cada cliente; saber quando deve ser feita a adaptação, tentando prever futuras oscilações na qualidade da rede; ou ainda a própria adaptação que poderá ser feita utilizando diferentes algoritmos de aproximação ou estabilização. Através da implementação modular, torna-se mais fácil o controlo os valores devolvidos pelo módulo, tornando possível a identificação e substituição rápida de componentes.

Nesse sentido, procedeu-se à identificação dos diferentes pontos-chave associados à solução, reservando um componente do módulo para cada um deles. Assim, foram

identificadas quatro divisões diferentes a aplicar na solução final, à qual foi atribuído o nome WSSBRA (Windless Streaming Server - Bit Rate Adaptation):

- Core (núcleo do módulo);
- Monitorização;
- Estimação;
- Adaptação.

Nas secções seguintes, são descritos e justificados os diferentes algoritmos cuja organização e comunicação estão representadas na **Figura 4-1**.

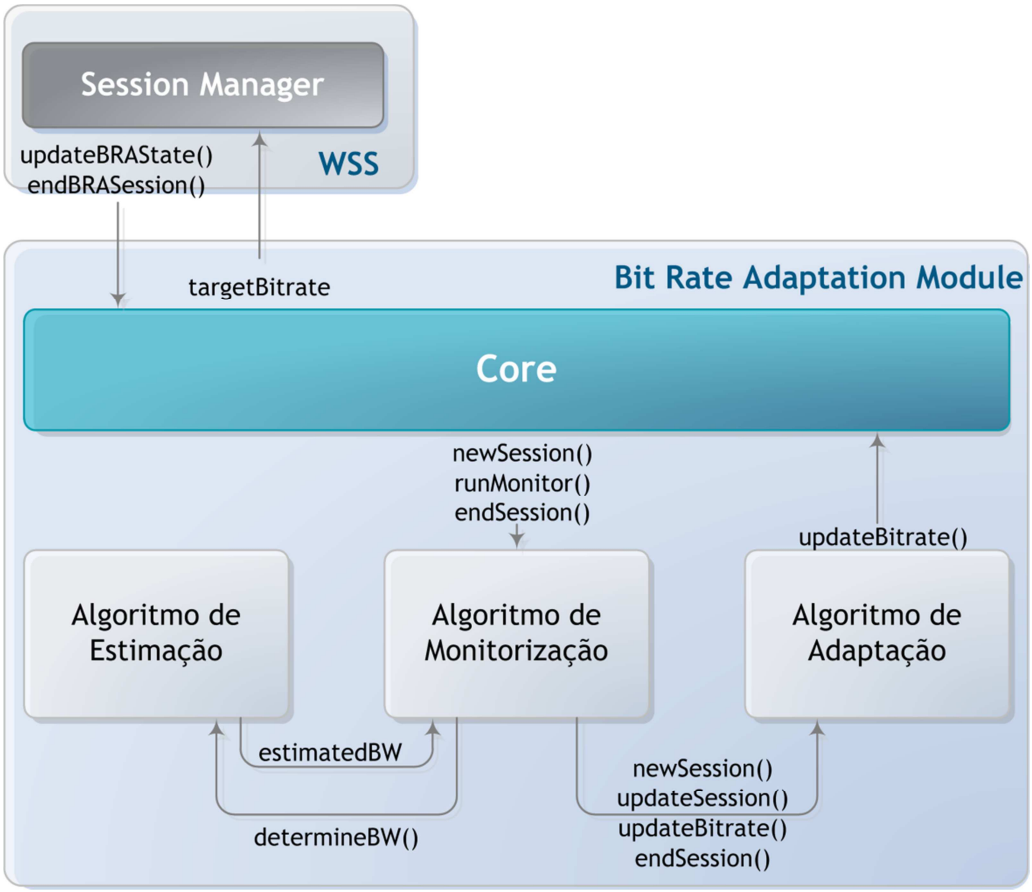

Figura 4-1 – Arquitectura do módulo WSSBRA.

4.1.1 - Entidades do WSS

1. SMAN (Session Manager)

Este elemento representa o *core* do WSS v3.0. As suas funções incluem a gestão de todas as sessões, dos recursos de DSPs (Digital Signal Processor) que realizam operações de adaptação, clientes RTSP para disponibilização de conteúdos externos, interface com sistemas externos de gestão de negócio, controlo de distribuidores de Live Feeds. O SMAN é também responsável pela agregação das estatísticas dos diferentes módulos que disponibiliza a uma interface *web* de operação e manutenção do WSS. Também lhe cabe a

responsabilidade do envio de *Traps* SNMP (Simple Network Management Protocol) para notificação de erros ou eventos.

4.1.2 - Entidades do módulo

1. Core

Componente principal do módulo. Implementa a API de comunicação com o SMAN e regista num histórico com tamanho editável os dados enviados pelo mesmo.

2. Algoritmo de monitorização

Este algoritmo tem como funcionalidade a monitorização dos valores recebidos do SMAN, e a consequente decisão de adaptação ou não do *bit rate* utilizado na codificação do vídeo.

3. Algoritmo de estimação da largura de banda

Determina e devolve a estimativa da largura de banda utilizada de uma dada sessão, utilizando nos cálculos os valores das variáveis:

- sessionId - Identifica a sessão para a qual os cálculos estão a ser efectuados;
- RTPsequenceNumber - Indica o número de sequência do último pacote RTP recebido pelo cliente;
- numberLostPackets - Número total de pacotes perdidos pelo cliente;
- receptionTimestamp - Timestamp da recepção do relatório RTCP do receptor;
- bytesSent - Valor do contador de *bytes* enviados.

4. Algoritmo de adaptação

Recebe do algoritmo de monitorização a estimativa da largura de banda e o identificador de sessão. Quando solicitado, devolve o valor do *bit rate* que o codificador deve utilizar na codificação do vídeo a enviar para o cliente.

4.1.3 - Configuração

Visto que os parâmetros do módulo poderão ter de ser alterados dependendo de inúmeros factores como, por exemplo, o tipo de conteúdos a transmitir, ou ainda a prevenção de falhas na rede, foi criado um único ficheiro de configuração, com o nome: “bra.conf”. Neste ficheiro, existem duas secções distintas, a configuração do registo onde se definem as variáveis associadas aos *Logs* gerados (nome, *path*, tipo de registo) e uma segunda secção que define as variáveis do módulo e respectivos algoritmos. Este ficheiro tem um formato definido numa biblioteca desenvolvida na PT Inovação, o WLOG. De seguida são apresentadas algumas linhas desse ficheiro:

```
################################
# Logging Section              #
################################
[Logging]
log.global.output=file:wssbraLog.log
log.main.all=1
log.main.error=1
log.test.info=1
log.test.debug=0
log.main.name=adaptMod

################################
# Variables Section            #
################################
[Core]
algorithms_path = /opt/ptin/wss/conf/

[Monitor]
jitter_threeshold=0.15
jitter_threeshold_crit=0.7
upper_limit=500

[Adaptor]
maxBitrate=600000
minDiff=10000
```

4.1.4 - Diagrama de sequência do módulo

É apresentado na **Figura 4-2** o diagrama de sequência do módulo WSSBRA.

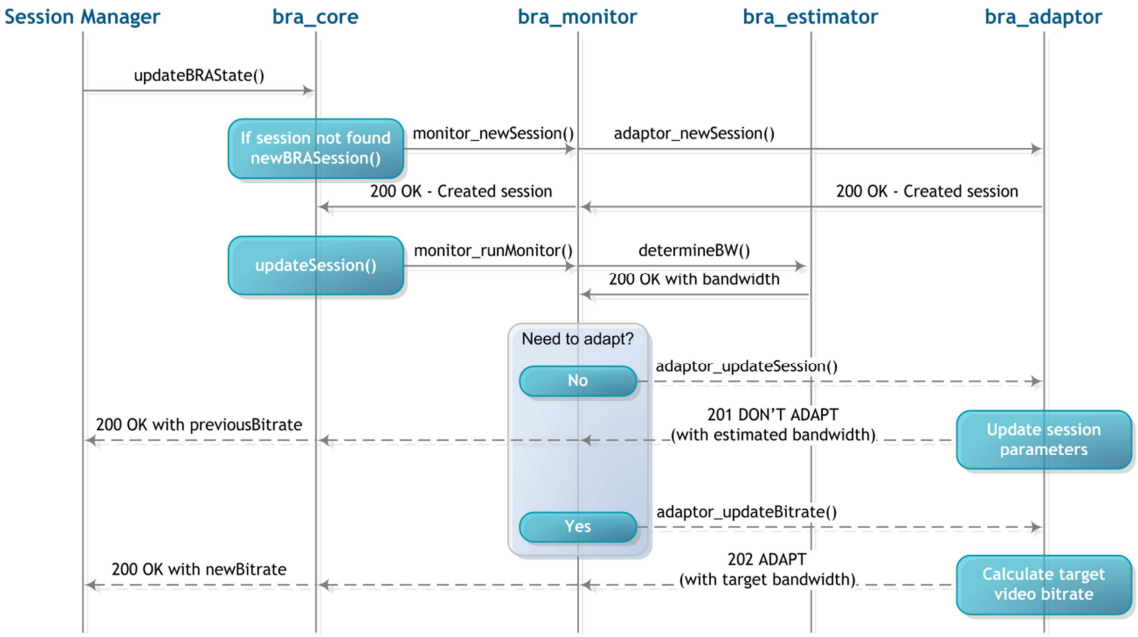

Figura 4-2 – Diagrama de sequência do módulo WSSBRA.

4.2 - Componente core (bra_core.c)

Este é o núcleo do módulo de adaptação. Mesmo que não existam algoritmos de monitorização, adaptação ou estimação disponíveis, este componente possibilita que o módulo receba e registe informações sobre as sessões activas. Nesse caso não existirá adaptação enquanto não existirem algoritmos para o efeito. A principal função deste componente é a comunicação com o SMAN; para tal, foi definida uma API.

Na **Figura 4-3** está representado o diagrama de sequência deste componente.

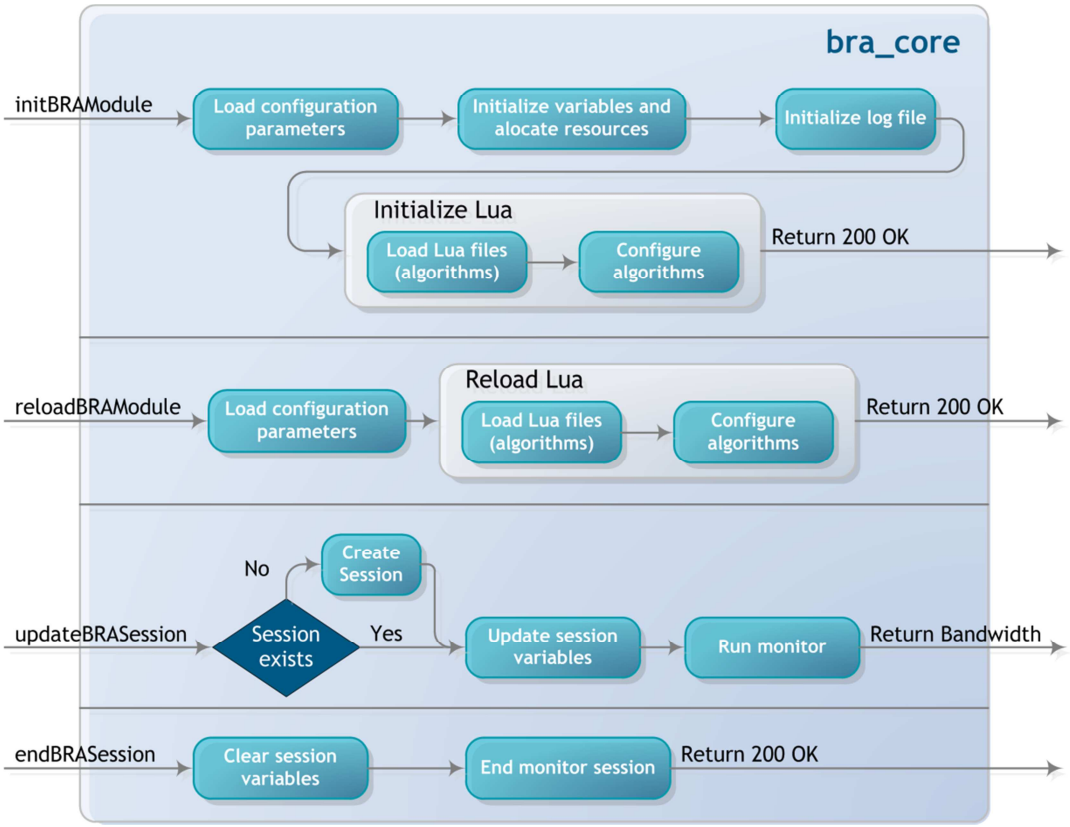

Figura 4-3 – Diagrama de sequência do componente Core do WSSBRA.

Interface com SMAN

1. **initBRAModule** (maxSessions)

Função de inicialização do módulo. Esta função abre e executa os algoritmos escritos em Lua, e aloca os recursos para um dado número de sessões máximas.

2. **reloadBRAmodule** ( )

Permite que os algoritmos Lua sejam actualizados sem ser necessário parar a execução do módulo de adaptação.

3. **updateBRAState** (sessionId, RTPSequenceNumber, jitter, numberLostPackets, fractionLost, bytesSent)

Função que recebe os valores do SMAN, actualiza o seu histórico, e invoca a função de monitorização. Depois de receber do algoritmo de monitorização o valor do *bit rate* com que o vídeo deve ser codificado, retorna o mesmo para o SMAN. Caso a sessão de destino ainda não exista, ou tenha sido apagada, é iniciada uma nova sessão.

4. **endBRASession** (sessionId)

Finaliza uma dada sessão.

Este componente possui ainda algumas funções que efectuam a troca de informação com os algoritmos desenvolvidos em Lua. Estas funções possibilitam, por exemplo, que os algoritmos acedam aos valores guardados no histórico do *core*, recolham parâmetros de configuração, ou ainda registem mensagens com formato pré-definido nos *logs*.

A função seguinte é um exemplo da comunicação dinâmica entre as linguagens C e Lua. Esta função, invocada pelos algoritmos Lua, tem como objectivo devolver parâmetros definidos no ficheiro de configuração. Este passo intermédio entre os algoritmos e a leitura do ficheiro de configuração revela-se necessário, pois o bra_core utiliza uma função definida no *framework* WOSX para ler ficheiros de configuração que seguem um modelo definido noutra biblioteca, o WLOG, como é o caso do WSSBRA.

Esta função tem a particularidade de ser completamente dinâmica e geral, ou seja, funciona para qualquer algoritmo Lua, para qualquer variável definida no ficheiro de configuração, e pode devolver quantas variáveis o algoritmo pedir. É invocada utilizando o seguinte código Lua:

```
//varA, varB = ("caller", "varA", "varB")
//caller -> section where the variable is placed in the config file, example: "Monitor",
"Adaptor"
jitter, maxBitrate = getConfParams("Monitor", "jitter", "maxBitrate")
```

De seguida está a definição da função:

```
static int getConfParams(lua_State *L){
   /* get number of arguments */
   int n = lua_gettop(L);
   char *caller = lua_tostring(L,1);
   char *params[n];
   char var_tmp[32];
   /* loop through each argument */
   for (int i = 2; i <= n; i++)
   {
      params[i-1] = lua_tostring(L, i);
      InoGetConfigString(caller, params[i-1], "0",var_tmp, 32, s_braCfgFile);
      lua_pushnumber(L, atof(var_tmp));
   }
   /* return the number of results */
   return n-1;
}
```

4.3 - Algoritmo de monitorização (bra_monitor.lua)

Na **Figura 4-4** está representado o diagrama de sequência deste componente.

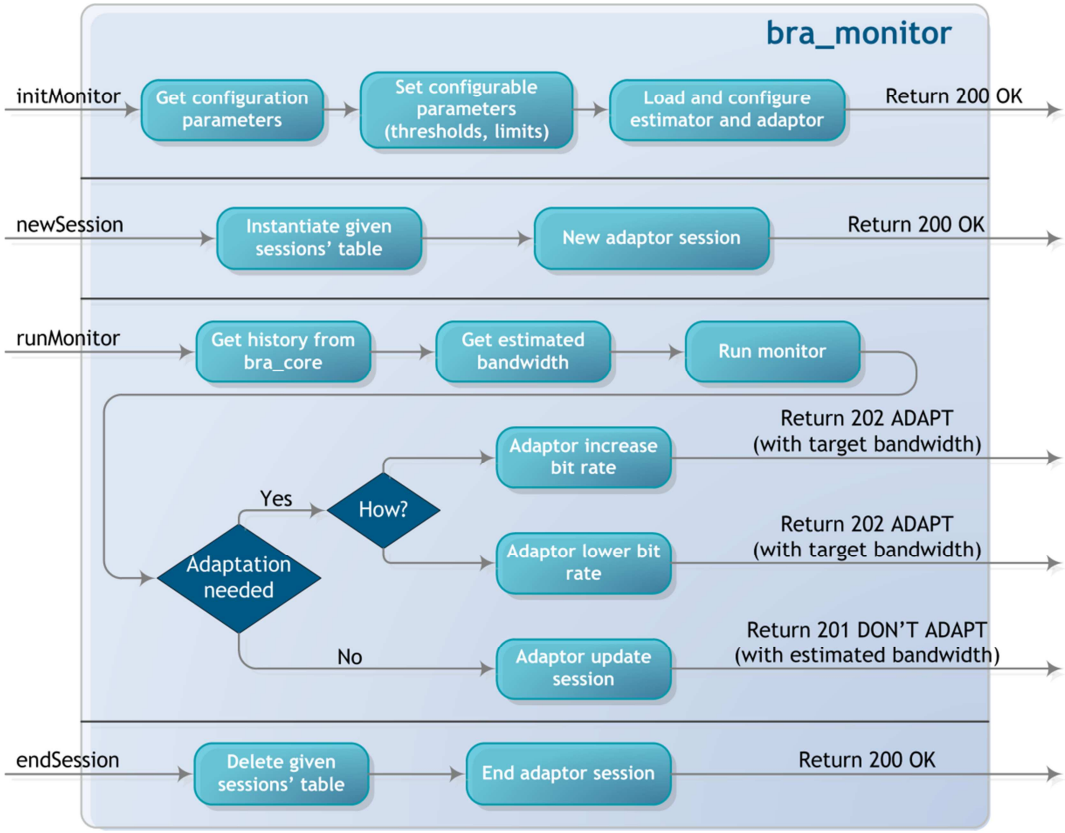

Figura 4-4 – Diagrama de sequência do algoritmo monitor do WSSBRA.

Como o próprio nome indica, este algoritmo tem como função monitorizar os dados recebidos do SMAN, seguindo algumas condições configuráveis (variação do *jitter*, número máximo de pacotes perdidos, entre outros), para decidir sobre a necessidade de alteração do *bit rate* utilizado na codificação do vídeo.

Quando recebe um pedido, o algoritmo invoca uma função do *core* que devolve os valores contidos no histórico do mesmo, sendo que o parâmetro mais importante para este algoritmo é o valor do *interarrival jitter*, uma vez que este dá indicações sobre o congestionamento da rede, podendo mesmo prever perda de pacotes. Outros valores como o número de pacotes perdidos ou os valores anteriores do *bit rate* do vídeo enviado para o cliente são também utilizados nas verificações, tendo, no entanto menos peso na decisão.

De seguida está apresentado o código onde é efectuada a decisão pelo algoritmo de monitorização relativamente à necessidade de adaptação:

```
bandwidth = estimatorDetermineBW( clientId )
if (jitter>meanjitter × 6 and meanjitter~=0) then
    instruction=3
elseif (jitter>meanjitter × 3 and meanjitter~=0) then
    instruction=2
elseif (LostPackets/SentPackets>0.2) then
    instruction=5
elseif (LostPackets/SentPackets>0.1) then
    instruction=4
else
    instruction=1
end
if (jitter<=meanjitter × 3) then
    updateMeanJitter()
    passedCounter=0
else
    passedCounter+=1
end
if (passedcounter > 5) then
    updateMeanJitter()
end
if (passedcounter > 20 or countjitter>=100) then
    restartMeanJitter()
end
adaptorUpdateBitrate(clientId, bandwidth, instruction)
```

Os valores dos limiares como a variação do *jitter* ou o número de pacotes perdidos que levam à adaptação são configuráveis através do ficheiro de adaptação, e foram obtidos através da análise de testes realizados ao algoritmo. O tipo de testes foi semelhante aos descritos nesta dissertação na 4.6.2 -. Para obter os valores dos limiares, foram analisadas as variações do *jitter* que levaram a perdas significativas de pacotes.

Para além da monitorização do *interarrival jitter*, é ainda verificado o número de pacotes perdidos. Esta verificação revelou-se necessária, pois existem reprodutores de multimédia que não enviam o valor do *jitter* nos relatórios RTCP, nomeadamente o Quicktime Player. Assim, mesmo que não seja possível obter o *jitter*, o algoritmo detecta a necessidade de adaptação, mas mais tardiamente. Ou seja, nessa situação é reactivo e não preventivo.

4.4 - Algoritmo de estimação (bra_estimator.lua)

Este algoritmo tem a importante função de, através das informações reduzidas enviadas pela aplicação cliente, calcular a velocidade a que os dados estão a chegar ao cliente. Este método de cálculo tem associados arredondamentos que introduzem erros incontornáveis.

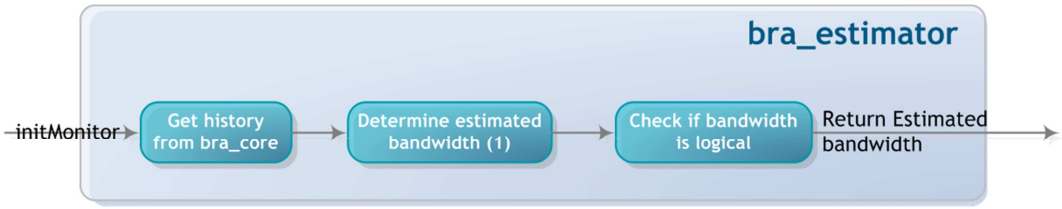

Figura 4-5 – Diagrama de sequência do algoritmo de estimação do WSSBRA.

Na **Figura 4-5** está representado o diagrama de sequência deste componente e, identificado com (1) está o passo que o algoritmo apresentado a seguir é executado.

$$timeInterval = (receptionTimestamp_f - receptionTimestamp_i) / 1000$$
$$totalSentBytes = bytesSent_f - bytesSent_i$$
$$nPacketsSent = RTPSequenceNumber_f - RTPSequenceNumber_i$$
$$totalLostPackets = numberLostPackets_f - numberLostPackets_i$$
$$medSize = totalSentBytes / nPacketsSent$$
$$nBytesReceived = totalSentBytes - totalLostPackets \times medSize$$
$$estimatedbandwidth = nBytesReceived \times 8 / diffTime$$

4.5 - Algoritmo de adaptação (bra_adaptor.lua)

Este componente é responsável pela definição do novo *bit rate* com que o vídeo enviado para o terminal do cliente deve ser codificado. Caso o algoritmo de monitorização detecte que não é necessário ou mesmo possível efectuar uma alteração de *bit rate* (por exemplo, o limite máximo ter sido atingido), apenas indica ao algoritmo de adaptação para actualizar as variáveis da respectiva sessão; caso contrário, se o algoritmo de monitorização detectar problemas na transmissão invoca uma função que recebe como argumentos o identificador da sessão, a largura de banda estimada pelo algoritmo de estimação, e ainda a instrução que indica qual o tipo de adaptação necessário. De seguida determina qual o *bit rate* a enviar para o cliente aplicando diferentes alterações na largura de banda estimada consoante a instrução recebida.

Para evitar possíveis erros provenientes de um decréscimo do *bit rate* seguido de um aumento imediato do mesmo, foi implementado um contador que coloca o módulo de adaptação em prevenção, impedindo-o de aumentar o *bit rate*. O valor do contador é relativo a cada instrução, isto é, adaptações mais bruscas originam maior tempo de prevenção.

Foram definidas cinco instruções com a relação causa-efeito indicada na **Tabela 4-1**.

Tabela 4-1 — Relação causa-efeito para os algoritmos de monitorização e adaptação.

Instrução	Causa	Consequência
1	Condições de rede favoráveis ao aumento de qualidade dos conteúdos transmitidos	Aumentar o *bit rate* para 1.1 × largura de banda estimada
2	Jitter superior a 3 vezes a média, diminuir o *bit rate* como prevenção	Diminuir o *bit rate* para 0.9 × largura de banda estimada Iniciar prevenção contra aumento de *bit rate* durante 3 iterações
3	Jitter superior a 6 vezes a média, rede em estado crítico, diminuição significativa do *bit rate*	Diminuir o *bit rate* para 0.7 × largura de banda estimada Iniciar prevenção contra aumento de *bit rate* durante 6 iterações
4	Perdidos 10% dos pacotes enviados, reagir com pequena diminuição do *bit rate*	Diminuir o *bit rate* para 0.9 × largura de banda estimada Iniciar prevenção contra aumento de *bit rate* durante 3 iterações
5	Perdidos 20% dos pacotes enviados, transmissão muito degradada	Diminuir o *bit rate* para 0.8 × largura de banda estimada Iniciar prevenção contra aumento de *bit rate* durante 6 iterações

Foi ainda implementado outro algoritmo de adaptação, onde era utilizado um controlador PID (Proportional Integrative Derivative controller) para determinar o valor do *bit rate* a enviar para um dado cliente. Essa solução apresentava melhorias na estabilidade do *bit rate* utilizado na codificação, pois suavizava as adaptações e, como mantinha um histórico dos erros ocorridos, possuía maior resistência a erros. No entanto, foi abandonada devido à sua lenta resposta relativamente à apresentada anteriormente.

4.6 - Testes

Nesta secção são apresentados os testes realizados aos algoritmos de estimação e monitorização individualmente. Os resultados da aplicação do algoritmo de adaptação serão analisados na secção seguinte onde a solução desenvolvida é testada no seu todo.

4.6.1 - Algoritmo de estimação

No caso deste algoritmo, os testes são meramente objectivos. A sua única função é, com base nas informações contidas nos relatórios RTCP, calcular a estimativa da largura de banda utilizada. Por outras palavras, este algoritmo retorna a velocidade com que a aplicação cliente recebe os conteúdos. Assim, os testes realizados consistem na verificação dos valores obtidos a partir deste algoritmo, comparando-os com os valores lidos no cliente.

Para registar os valores reais da velocidade de recepção dos dados no cliente foi utilizada a ferramenta Wireshark[32], a qual analisa o tráfego da rede, registando todos os pacotes

recebidos e enviados por uma determinada interface durante toda a duração da análise. Esta ferramenta permite a exportação dos dados recolhidos com um intervalo de tempo configurável para ficheiros de diferentes formatos.

Cenário de testes:

- Vídeo codificado com *bit rate* constante de 800 kbit/s;
- Ferramenta wireshark a registar o tráfego de 1 em 1 segundo;
- Acesso com portátil a uma rede Wi-Fi com ligação máxima de 56 mbps;
- WSSBRA configurado para registar valores das variáveis de 1 em 1 segundo num ficheiro CSV (Comma-Separated Values);
- Acesso realizado através do reprodutor VLC media player, com envio de relatórios RTCP de 2 em 2 segundos, e gravação do vídeo para ficheiro local;
- Movimentação durante as sessões com afastamento do *wireless access point* de aproximadamente 20 metros com paredes no meio;
- 4 acessos realizados (os 10 primeiros segundos foram descartados sendo apenas considerados os 70 segundos seguintes).

Depois de registados os dados de todas as sessões, foram obtidos os gráficos apresentados na **Figura 4-6** e respectivos erros médios.

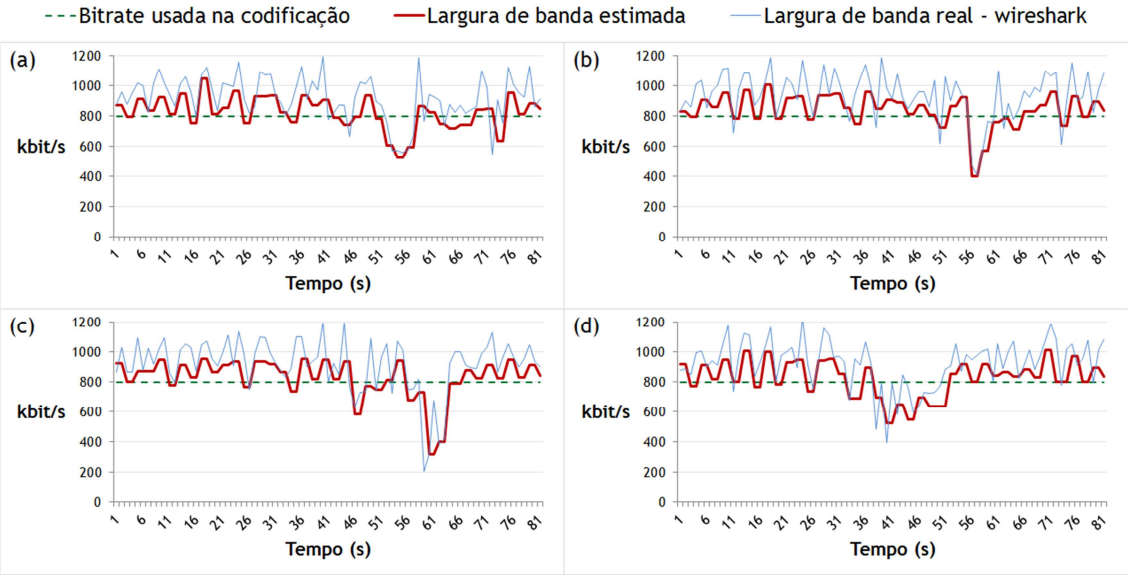

Figura 4-6 – Gráficos obtidos nos testes ao algoritmo de estimação, com erros médios: (a) 12,24 %; (b) 12,08%; (c) 14,60%; (d) 13,04%.

Para N igual ao número de valores registados, eBW a largura devolvida pelo algoritmo e rBW pelo wireshark, os valores do erro foram obtidos através da fórmula

$$\sum_{i=1}^{N}\left(\frac{|rBW(i)-eBW(i)|}{rBW(i)}\right)\times\frac{100}{N} \qquad \textbf{Equação 4-1}$$

Um dos factores que contribui para que existam erros na estimação é o facto de existirem mais dados a chegar ao cliente para além dos enviados pelo servidor de *streaming*. Visto que o estimador apenas considera os *bytes* enviados pelo WSS desconhecendo por completo o número total que realmente é recebido pelo cliente, existem sempre mais dados a chegar ao cliente para além dos enviados pelo WSS. O que significa que o valor da largura de banda utilizada real será normalmente superior ao estimado pelo algoritmo.

Uma vez que este algoritmo devolve a largura de banda utilizada pelo cliente e não a total disponível, existem algumas condicionantes na sua utilização. Quando ocorrer degradação da ligação, ou seja, quando a velocidade de recepção de dados for inferior ao *bit rate* utilizado na codificação dos conteúdos transmitidos, o terminal receptor terá de utilizar toda a sua largura de banda disponível para conseguir receber os dados. Ou seja, em caso de degradação, o valor estimado corresponde realmente ao valor da largura de banda disponível para o utilizador. Caso a qualidade da ligação permita a recepção sem problemas por parte do terminal receptor, o mesmo apenas utilizará a largura de banda necessária para transmitir todos os dados, não atingindo o valor realmente disponível. Este facto foi comprovado através do aumento do *bit rate* para valores superiores aos registados tanto pelo algoritmo de estimação, como pela ferramenta Wireshark, o que originou um aumento na largura de banda estimada não implicando erros na transmissão. Assim, para reduções de *bit rate*, é possível utilizar o valor estimado como limite máximo. No entanto, nas subidas deverá ser utilizado um valor superior ao estimado. Este comportamento poderá originar alguma instabilidade, portanto revela-se necessário definir limites máximos que o valor do *bit rate* poderá assumir.

Através dos resultados obtidos, é possível concluir que os valores devolvidos pelo algoritmo apresentam um erro médio na ordem dos 13% relativamente aos valores registados pela ferramenta Wireshark. Tendo em conta todas as condicionantes no seu cálculo (informações reduzidas do cliente, assíncronismo no envio de relatórios RTCP, arredondamento no tamanho médio de pacotes, tráfego na rede, entre outros), o valor da estimativa obtido é uma boa aproximação ao real, podendo por isso ser utilizado na adaptação sem grandes implicações no resultado final.

4.6.2 - Algoritmo de monitorização

Espera-se que este algoritmo seja capaz de detectar problemas na rede, indicando ao algoritmo de adaptação que deve alterar o *bit rate* com que o vídeo é codificado. Não só deve detectar problemas na rede, como também tem a função de indicar ao algoritmo de adaptação que pode aumentar o *bit rate* se a rede tiver condições para tal.

O algoritmo de monitorização envia as suas indicações através de uma variável que pode possuir os seguintes valores:

1. Rede em estado normal;
2. *Jitter* superior a 3 vezes a média, prevê degradação da rede, realizar pequena adaptação;
3. *Jitter* superior a 6 vezes a média, grande degradação da rede, realizar adaptação significativa;
4. 10% pacotes perdidos, realizar pequena adaptação;
5. 20% pacotes perdidos, degradação da rede, realizar adaptação significativa.

Cenário de testes:

- Vídeo codificado com *bit rate* constante de 800 kbit/s;
- Acesso com portátil a uma rede Wi-Fi com ligação máxima de 56 mbps;
- WSSBRA configurado para registar valores das variáveis de 1 em 1 segundo num ficheiro CSV;
- Acesso realizado através do reprodutor VLC media player, com envio de relatórios RTCP de 2 em 2 segundos, e gravação do vídeo para ficheiro local;
- Movimentação durante as sessões com afastamento do *wireless access point* de aproximadamente 20 metros com paredes no meio;
- 4 acessos realizados (os 10 primeiros segundos foram descartados sendo apenas considerados os 70 segundos seguintes).

Depois de registados os dados, foram obtidos os gráficos apresentados na **Figura 4-7**.

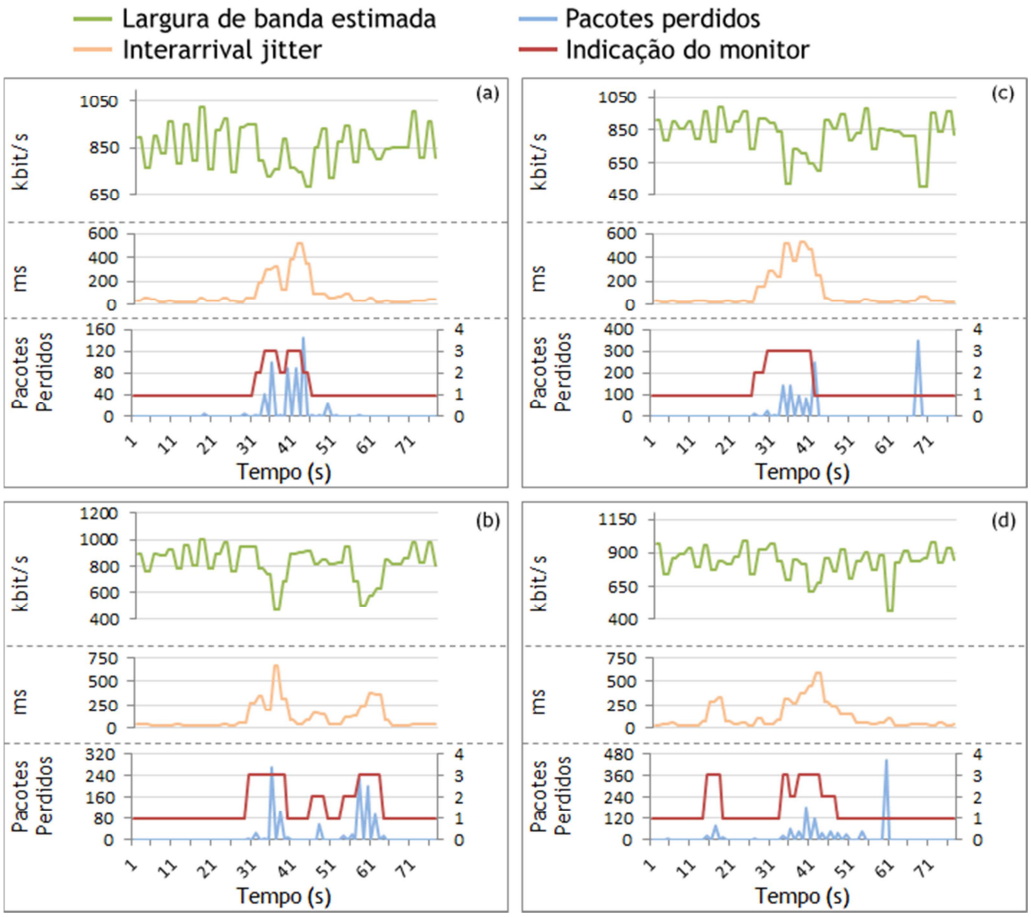

Figura 4-7 – Gráficos obtidos nos testes ao algoritmo de monitorização.

Analisando o terceiro gráfico de cada recolha que representa as indicações do algoritmo de monitorização juntamente com o número de pacotes perdidos, podemos confirmar que o algoritmo desenvolvido apresenta alguma predição relativamente à perda de pacotes. Esta propriedade é bem visível, por exemplo, aos 30 segundos da recolha (b) onde o algoritmo de

monitorização indica que deve ser feita uma adaptação e, passado 2 segundos começam a ocorrer perdas significativas e constantes de pacotes.

Para além da predição que apresenta, é possível verificar que o algoritmo é resistente a erros nomeadamente aos 70 segundos no caso (b) ou 60 no caso (d). Nestes pontos ocorrem picos no valor do número de pacotes perdidos sem que existam variações elevadas no *jitter*. Este facto pode indicar que ocorreu um erro na recepção do relatório RTCP ou até mesmo na interpretação dos dados por parte do sistema. Nestes dois casos, o algoritmo de monitorização não reage, eliminando possíveis erros na adaptação.

Outro factor importante a mencionar é a coerência da monitorização uma vez que, sempre que ocorreram perdas significativas de pacotes, foram detectadas e mesmo previstas pelo algorimo.

4.7 - Validação da solução - *Bit Rate Adaptation* vs *Constant Bit Rate*

Depois de realizados testes independentes aos algoritmos de estimação e monitorização, resta verificar o funcionamento do algoritmo de adaptação. Visto que este é dependente dos anteriores, os testes foram realizados não só ao algoritmo de adaptação, mas a todo o módulo desenvolvido.

Nesse sentido, foram recolhidos resultados tanto de sessões com adaptação de *bit rate* como de sessões com conteúdos codificados com *bit rates* constantes para possibilitar a comparação entre os mesmos. Como o objectivo desta dissertação é melhorar a QoE do utilizador em ambientes móveis com ligação degradada, os acessos foram realizados a partir de um terminal móvel, Nokia N95 através da aplicação Realplayer, utilizando a rede HSDPA (High Speed Downlink Packet Access).

Ao contrário dos anteriores testes realizados, para o módulo de adaptação foram feitos testes objectivos e subjectivos. Com esta abordagem espera-se que seja possível perceber de que modo as esperadas melhorias objectivas se reflectem na opinião subjectiva de cada utilizador.

4.7.1 - Análise objectiva

Apesar de várias tentativas para obter as mesmas condições de rede para diferentes sessões, nomeadamente entre duas sessões com e sem adaptação, os resultados obtidos não permitiram retirar conclusões objectivas, pois uma das sessões saía sempre prejudicada. Assim, a opção foi definir diferentes cenários de teste, com o objectivo de simular diferentes condições de rede.

Foram efectuadas seis recolhas, três com adaptação e três com *bit rate* constante. Para simular diferentes condições de rede, cada sessão foi configurada com parâmetros diferentes. Para as sessões com *bit rate* constante, foram definidos três *bit rates* a utilizar na

codificação: 300, 450 e 600 kbit/s. Para as sessões com adaptação, foram alterados os valores máximos que o *bit rate* poderia assumir: 400, 550 e 700 kbit/s, ou seja, sempre 100 kbit/s possíveis acima das sessões sem adaptação.

Depois de definidos os parâmetros para cada sessão, foi seguido um percurso definido previamente sempre no interior de um edifício. O percurso iniciou-se com a movimentação de aproximadamente 20 metros passando por diferentes divisões, seguindo-se a entrada num local isolado (escadas com isolamento) e terminou com o retorno ao local de origem. A movimentação foi constante durante toda a duração do vídeo.

Na **Figura 4-8** estão apresentados os gráficos obtidos dos diferentes testes.

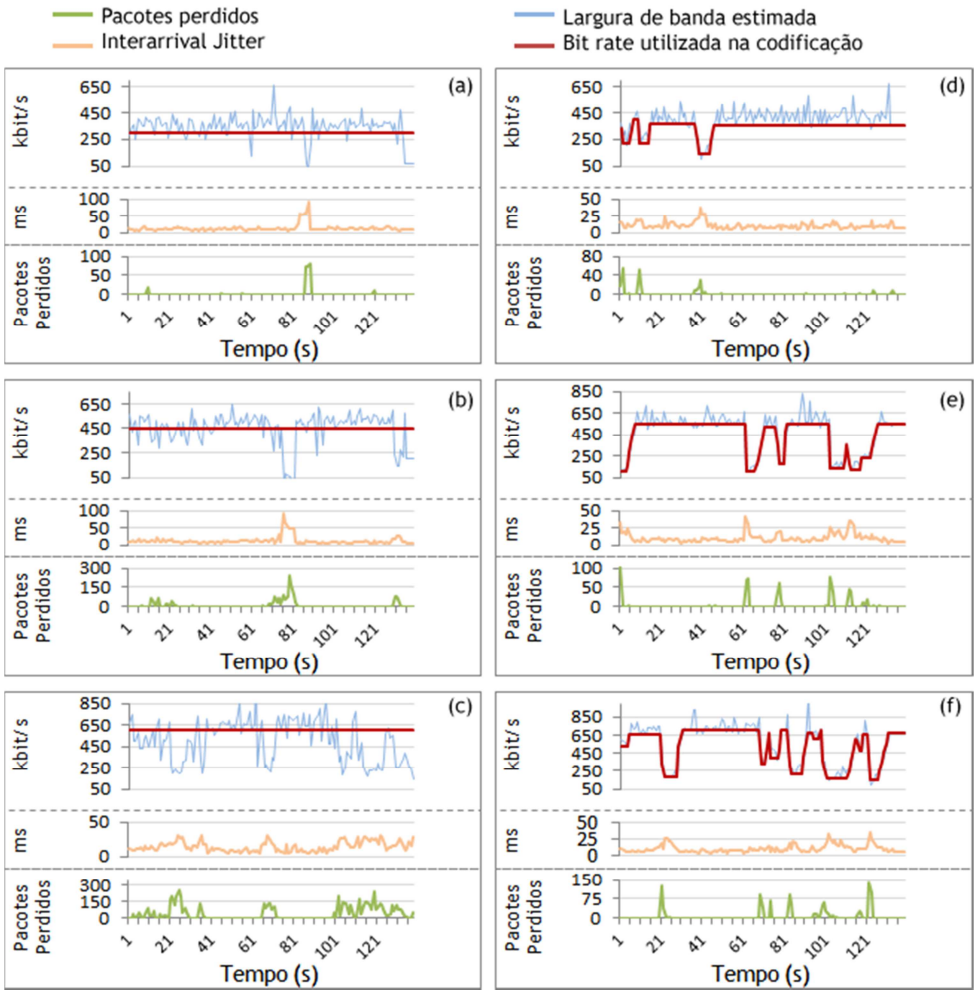

Figura 4-8 – Sessões com *bit rate* constante: (a)300 kbit/s, (b)450 kbit/s e (c)600 kbit/s; Sessões com *bit rate* variável (limites máximos): (a)400 kbit/s, (b)550 kbit/s e (c)700 kbit/s.

Analisando os gráficos, podemos verificar e confirmar que as perdas de pacotes mais significativas têm relação com o aumento do valor do *interarrival jitter*. É também possível confirmar o funcionamento do algoritmo de estimação, pois quando ocorrem perdas de pacotes, a largura de banda estimada está abaixo do *bit rate* utilizado na codificação do vídeo.

Uma das melhorias visíveis graficamente é a recuperação à ocorrência de erros que o módulo introduz no sistema. Analisando os gráficos (b) e (e) aos 81 segundos, verificamos que a sessão que possui adaptação demora perto de dois segundos a eliminar as perdas de pacotes enquanto que no caso da sessão (b), os erros prolongam-se continuamente por aproximadamente 10 segundos. Esta característica é visível também nos gráficos (c) e (f) onde ocorrem muitos erros nas transmissões a partir dos 100 segundos. No caso da sessão (f), são rapidamente anulados pela adaptação realizada.

Este comportamento origina menos pacotes perdidos nas sessões que possuem adaptação, tornando as transmissões mais eficientes e optimizando a utilização de recursos de rede. Na **Tabela 4-2** são apresentados os valores obtidos nos testes realizados.

Tabela 4-2 — Resultados dos testes realizados ao módulo de adaptação.

	Sem adaptação			**Com adaptação**		
	(a)	**(b)**	**(c)**	**(d)**	**(e)**	**(f)**
Média do *bit rate* utilizado na codificação (kbit/s)	300	450	600	336,4	441,8	521,4
Média da estimativa da largura de banda utilizada (kbit/s)	342,6	448,9	501,8	394,9	468,5	552,8
Total de pacotes enviados	13882	19868	24322	19610	17918	21868
Pacotes perdidos (%)	1,9	8	14,4	1,2	3,4	4,8
***Interarrival jitter* médio (ms)**	13,4	13,1	14,7	10,8	10,6	10,7

Através da análise da anterior tabela, é possível verificar que objectivamente, a solução desenvolvida apresenta melhorias significativas na eficiência da transmissão.

Desde logo podemos verificar uma diminuição global no número de pacotes perdidos. Podemos também concluir que a redução de pacotes perdidos é tanto maior quanto maior for o *bit rate* dos conteúdos multimédia enviados. É interessante verificar que no caso das sessões (a) e (d), apesar do *bit rate* utilizado na codificação ser maior na sessão com adaptação, o número de pacotes perdidos é menor.

Outro indicador da melhoria obtida com a aplicação do módulo é o valor médio do *interarrival jitter*. Este campo, como explicado na 3.6 - deste documento, indica os atrasos ocorridos na entrega de pacotes RTP. Ou seja, quanto maior for o seu valor, mais atrasos na transmissão ocorrem e consequentemente mais erros na reprodução. De realçar que o aumento verificado na sessão (c) relativamente às restantes sessões com *bit rate* constante, não se verifica na sessão (f) relativamente a (d) e (e).

4.7.2 - Análise subjectiva

Analisadas as melhorias objectivas da aplicação do módulo de adaptação, resta saber se as mesmas se manifestam na qualidade subjectiva dos conteúdos multimédia.

Para realizar a avaliação subjectiva foram efectuados dois acessos distintos com um computador portátil com acesso à rede HSDPA possibilitado por uma *pen* 3G. Durante a sessão, foi percorrido o percurso definido anteriormente na análise objectiva. Para registar visualmente a reprodução, foi utilizada uma ferramenta de captura de ecrã. Depois de possuir as gravações, foi utilizado um programa de edição de vídeo para sincronizá-las no seu início. De seguida foram colocadas lado a lado num mesmo vídeo e retirados apenas 30 segundos da reprodução. Na apresentação dos vídeos não existe qualquer referência a qual o vídeo que possui adaptação.

Depois de possuir o vídeo devidamente editado para visualização e comparação, o mesmo foi disponibilizado numa página *web* (http://websantos.net/bra) onde foi sujeito à avaliação por um grupo de 35 pessoas constituído por especialistas com experiência na transmissão de vídeo em ambientes móveis e pessoas externas ao meio. A avaliação foi feita de acordo com a escala MOS (Mean Opinion Score)([33], secção B.4.5).

Na **Figura 4-9** estão apresentados os gráficos relativos à totalidade das sessões para os dois vídeos avaliados. Em destaque estão os segmentos de vídeo mostrados aos participantes na votação.

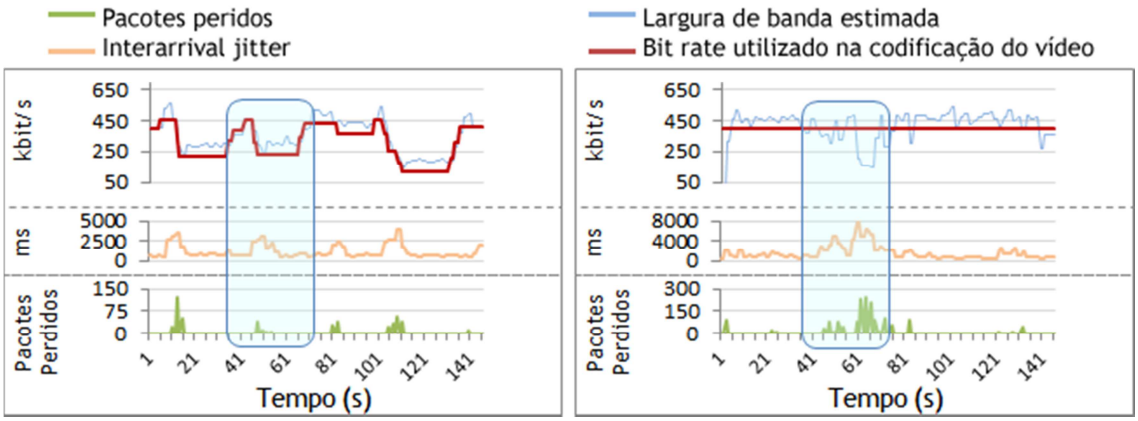

Figura 4-9 – Gráficos relativos aos vídeos avaliados subjectivamente.

Os resultados objectivos obtidos são apresentados na **Tabela 4-3**.

Tabela 4-3 — Resultados objectivos para os vídeos utilizados na avaliação subjectiva.

	Sem adaptação	Com adaptação
Média do *bit rate* utilizado na codificação (kbit/s)	400	301,9
Média da estimativa da largura de banda utilizada (kbit/s)	414,7	346,3
Total de pacotes enviados	17847	13906
Pacotes perdidos (%)	8,7	3,4
***Interarrival jitter* médio (ms)**	1635,4	1192,9

Como podemos verificar, objectivamente o vídeo que possui adaptação apresenta menores perdas e atrasos na transmissão confirmando os resultados obtidos previamente em diferentes testes.

Os resultados obtidos nas classificações subjectivas dos dois vídeos estão presentes na **Tabela 4-4**.

Tabela 4-4 — Resultados da avaliação subjectiva realizada por 35 participantes.

	Com adaptação	Sem adaptação
1	0	7
2	2	22
3	13	6
4	20	0
5	0	0
Média	3,5	2,0

Juntamente com a classificação aos dois vídeos apresentados, foi pedido aos participantes que deixassem um pequeno comentário com os aspectos que acharam mais relevantes na avaliação. De seguida são apresentados alguns dos comentários deixados por diferentes participantes. As referências aos vídeos 1 e 2 correspondem às situações com e sem adaptação respectivamente.

Eugene Fabian program manager

"2nd video seems to be slower and blurs a lot."

Eng. Nuno Ferreira

"O vídeo 1, na globalidade tem uma reprodução mais fluída, embora se notem muitos artefactos nomeadamente no início."

Eng. Pedro Pinto

"O vídeo 1 apresentou macroblocos durante a reprodução. O vídeo 2 apresentou ainda mais macroblocos bem como interrupções na reprodução."

Joana Relva

"Apesar de ocorrerem falhas em ambos os vídeos, o primeiro vídeo decorreu até ao fim sem interrupções maiores, enquanto que o segundo teve um intervalo de tempo em que a reprodução foi interrompida."

Prof.ª Dr.ª Maria Teresa Andrade

"Embora no vídeo 1 existam alguns erros que perturbam a visualização, a continuidade da sequência de imagens é assegurada, o que permite seguir bem o vídeo mesmo com um ou outro momento em que a imagem aparece bastante estragada. No vídeo 2, isso já não é verdade, uma vez que os erros duram mais tempo e a sequência natural das imagens não é perservada. Talvez a diferença entre os dois vídeos em termos de qualidade seja maior do que apenas um nível mas uma escala de 1 a 5 (que é em todo o caso a escala utilizada pela ITU no MOS) não permite muito mais (classifiquei o vídeo 1 com '3' e o vídeo 2 com '1')"

A partir dos resultados das classificações e dos comentários recebidos, é possível concluir que o vídeo com adaptação de *bit rate* representa melhorias tanto a nível de serviço (QoS) como na experiência proporcionada aos utilizadores (QoE). Um dos pontos essenciais foi detectado pelos participantes na votação, a diminuição das interrupções na reprodução. Como é visível no vídeo, no momento em que começa o ocorrer a degradação dos vídeos, o vídeo que não possui adaptação apresenta interrupções de vários segundos, enquanto que o vídeo com adaptação diminui a definição continuando a ser reproduzido. Apesar da diminuição significativa do *bit rate* com que o vídeo foi codificado (350 kbit/s no vídeo sem adaptação para 250 kbit/s no vídeo com adaptação), os participantes expectavelmente preferiram o vídeo com menor definição, mas sem interrupções na reprodução.

Nesse sentido, os objectivos propostos foram cumpridos. No entanto ainda existem bastantes melhorias possíveis até atingir um nível de adaptação que seja indistinguível para os utilizadores, uma vez que, apesar das classificações serem favoráveis à solução desenvolvida, houve algumas opiniões referentes a artefactos e degradação da imagem no vídeo com adaptação, nomeadamente os comentários do Eng. Nuno Ferreira e Eng. Pedro Pinto.

Capítulo 5

Conclusão

Neste capítulo é feito um balanço sobre todo trabalho realizado, com destaque para os objectivos que foram atingidos e o que poderia ser melhorado. São ainda expostas todas as dificuldades sentidas ao longo do trabalho e as lições aprendidas. O capítulo termina com uma abordagem ao trabalho futuro a ser realizado.

5.1 - Revisão do trabalho desenvolvido

Esta dissertação teve como objectivo principal o estudo e desenvolvimento de um módulo que possibilitasse melhorar a QoE do utilizador final de um sistema de *streaming*. Face aos resultados obtidos, tanto objectivos como subjectivos, conclui-se que o módulo cumpriu a sua função. No entanto, as melhorias visíveis nos valores da QoS, ou seja, no número de pacotes perdidos ou atrasos na entrega de pacotes não apresentam a mesma dimensão quando avaliadas subjectivamente. Enquanto que objectivamente foram observadas reduções de 50% no número total de pacotes de dados perdidos, reduções do atraso na entrega de pacotes na ordem dos 20%, subjectivamente ainda são visíveis demasiados artefactos no vídeo.

No entanto, se analisarmos individualmente os componentes de estimação e monitorização, os resultados são bastante positivos, sendo que na monitorização superam mesmo as expectactivas iniciais no sentido em que, em alguns momentos existe predição relativamente à perda de pacotes de dados. Tendo em conta que os valores contidos nos relatórios RTCP são reduzidos, e que se trata de uma transmissão sobre UDP onde não existem certezas na entrega de pacotes, e principalmente por não existir uma aplicação do lado do cliente, os erros obtidos na estimação e na monitorização são bastante aceitáveis.

Visto que o principal requisito da solução desenvolvida era a integração com o WSS - servidor de *streaming* da PT Inovação - deve ser mencionado o trabalho desenvolvido para que esta integração fosse possível. Foram utilizadas bibliotecas criadas por engenheiros da PT Inovação com o intuito de simplificar e padronizar a implementação e funcionamento dos vários sistemas. Funcionalidades como criação de threads, abertura e escrita em ficheiros, ou ainda a utilização de ficheiros de registo padronizados com funções de leitura automática de

variáveis foram utilizadas no módulo desenvolvido, possibilitando assim uma mais fácil integração com o sistema. Foram ainda implementadas funções que não acrescentam valor no âmbito desta dissertação, mas que no momento da integração se revelaram necessárias para que não ocorressem erros nas comunicações, e consequentes falhas críticas.

Outra característica do módulo implementado que se revelou bastante positiva foi a implementação dos algoritmos em linguagem Lua. Esta opção permitiu que fossem efectuadas alterações aos algoritmos desenvolvidos durante a realização de testes, sem ser necessário recompilar o código todo. Também foi bastante interessante o desenvolvimento das APIs de comunicação entre o componente core desenvolvido em linguagem C, e os algoritmos Lua. Foi um desafio bem superado, uma vez que foram implementadas funções genéricas que funcionariam sem problemas noutras implementações. Nomeadamente as funções que possibilitaram a recolha de resultados, e a configuração dos algoritmos *on the fly*, ou seja, sem ser necessário interromper a execução de todo o módulo.

Resta fazer uma comparação com as soluções descritas no levantamento do estado da arte inserido neste documento. Todas as outras soluções necessitam que o utilizador possua reprodutores multimédia específicos nos seus terminais. A solução implementada foi testada com diferentes terminais, sistemas operativos, e principalmente diferentes reprodutores multimédia tais como:

- Em Nokia, Android, Computadores portáteis e fixos, sistemas operativos Windows e Linux (várias distribuições);
- VLC media player em Windows, Linux e MacOS;
- Quicktime Player em Windows;
- Dogaplayer em Android.

Em todos eles, a solução desenvolvida teve sucesso na sua função. Nem sempre nas melhores condições, pois em alguns reprodutores os relatórios RTCP são enviados com intervalos de tempo demasiado espaçados chegando mesmo aos 5 segundos (o que implica atrasos de pelo menos 5 segundos na adaptação), mas em todos eles se verificaram melhorias objectivas nos resultados obtidos.

5.2 - Principais dificuldades

Uma das principais dificuldades sentidas no desenvolvimento da solução foi a obtenção de valores credíveis para os vários componentes, especialmente no algoritmo de estimação. Foi necessário efectuar um longo trabalho de investigação na tentativa de, a partir da quantidade muito reduzida de informação contida nos relatórios RTCP conseguir obter um valor que possibilitasse ter alguma noção do valor final do *bit rate* a utilizar. Foi mesmo necessário combinar informação recolhida no WSS nomeadamente o número de *bytes* enviados para o cliente, com informações contidas nos relatórios RTCP.

Para além das reduzidas informações, existiu ainda outra condicionante bastante significativa na utilização da informação contida nos relatórios RTCP. Apesar de o protocolo RTCP estar bem definido em RFC, os diferentes reprodutores multimédia utilizados

implementam o mesmo de formas diferentes. Por exemplo, o Quicktime Player não envia o valor do *interarrival jitter*, impossibilitando a previsão de erros da solução desenvolvida, obrigando-a a utilizar apenas o número de pacotes perdidos para efectuar as alterações. Também os valores enviados pelo reprodutor Dogaplayer para Android são bastante diferentes dos enviados pelo Realplayer, nomeadamente no valor do *jitter*. Enquanto os valores recebidos do Realplayer rondam sempre os 10-15 ms, no Dogaplayer são recebidos valores na ordem dos 4000/5000, indicando uma possível alteração nas unidades do parâmetro que deveriam ser em milisegundos como definido em RFC.

Para além dos valores das variáveis, também o intervalo entre o envio de relatórios RTCP difere entre reprodutores, assumindo valores de 1 segundo para o Realplayer, passando por 2 segundos no VLC media player até 5 segundos no Dogaplayer. Também a plataforma tem implicações nesta característica, por exemplo, a versão do Realplayer para Windows enviava relatórios de 2 em 2 segundos. Este facto afecta directamente o desempenho do módulo de adaptação, sendo que quanto maior for o intervalo entre o envio de relatórios, maior é o tempo de reacção do módulo e consequentemente, pior é o resultado final.

Finalmente, na realização de testes revelou-se bastante difícil garantir valores coerentes entre sessões, uma vez que a qualidade da rede difere muito de sessão para sessão. Apesar de ser possível obter comportamentos parecidos, existe sempre a justificação das sessões não terem exactamente a mesma qualidade de ligação, e por isso as respostas serem diferentes e incomparáveis.

5.3 - Possíveis melhorias

No sentido de melhorar a QoE do utilizador final num momento de adaptação de *bit rate*, foi identificado um problema relativo à mudança brusca do *bit rate* utilizado na codificação. Quando é detectada uma falha na rede e é feita a adaptação, se esta não coincidir com o fim da transmissão de um *frame* de vídeo, são perdidos pacotes, pois as placas de codificação fazem a transcodificação completa de um novo *I-frame*, ou seja, se a transmissão da imagem anterior for interrompida, a informação restante dessa imagem não chegará ao utilizador, provocando degradação na qualidade da imagem. Para resolver este problema, deveria ser implementada uma verificação adicional que apenas permitisse a adaptação quando um *frame* fosse totalmente transmitido ou apenas quando fossem enviados *I-frames*.

Outra melhoria relativa à adaptação seria a introdução de um histórico para os valores máximos de *bit rate* durante uma sessão para os quais a transmissão foi efectuada sem erros. Assim, o limite superior para o *bit rate* poderia ser dinâmico, e adaptado a cada sessão.

Finalmente, poderiam ser implementados métodos de codificação relativa ao conteúdo (*content-aware coding*), ou seja, para diferentes conteúdos com diferentes exigências de qualidade, seriam utilizados métodos com adaptações distintas. Por exemplo, em vídeos com muito movimento, seria utilizado um *bit rate* superior relativamente a vídeos com poucas transições e movimento. Outra solução mais simples e com menores requisitos de recursos seria realizar adaptação relativa ao contexto (*context-aware coding*), ou seja, poderiam ser

definidos tipos de programas para os quais deveria ser possibilitado maior *bit rate* como jogos de futebol onde existe a necessidade de ver com pormenor a localização da bola, e outros tipos como noticiários onde o mais importante não seja o vídeo, mas sim o áudio.

5.4 - Trabalho futuro

Após a realização deste trabalho e respectiva recolha e análise de resultados, torna-se evidente que ainda existe muito trabalho a fazer com o intuito de melhorar não só a QoS, mas principalmente a QoE do utilizador final. Nesse sentido, serão optimizados os algoritmos desenvolvidos, nomeadamente com a implementação de mais e melhores verificações relativas aos valores e limiares obtidos.

Serão também implementados diferentes algoritmos de adaptação, possibilitando a sua alteração em tempo-real para diferentes sessões.

Quanto à aplicação da solução desenvolvida, esta será inserida na versão 3.4 do servidor da PT Inovação, WSS. Para além disso, será ainda utilizada como objecto de estudo e exemplo em diferentes projectos sempre com o intuito de melhorar a qualidade da transmissão de conteúdos multimédia, como por exemplo, o projecto Europeu MEDIEVAL – MultimEDia transport for mobIlE Video AppLications.

Finalmente será implementado um monitor *web* que permitirá visualizar as diferenças entre vídeos com e sem adaptação, ao mesmo tempo em que se visualizam os gráficos relativos a essas mesmas sessões.

Referências

[1] Vantrix Corporation. “Bandwidth management and traffic shaping in wireless and broadband networks”. Technical report, 2009.
[2] Bubley Dean. “Mobile broadband computing - device market forecasts & business model scenarios”, Dezembro 2008. Disponível em http://www.disruptive-analysis.com/mobile_broadband.htm. Acesso em 16 de Junho de 2010.
[3] Inc Pyramid Research. “Mobile broadband for the masses - the case for bundled netbooks”, Maio 2009. Disponível em http://www.researchandmarkets.com/reports/993154. Acesso em 16 de Junho de 2010.
[4] AndroLib.com. “Distribution downloads in the android market”, Dezembro 2009. Disponível em http://www.androlib.com/appstatsdownloads.aspx. Acesso em 16 de Junho de 2010.
[5] European Telecommunications Standards Institute. “Etsi tr 102 783”, Março 2010. http://www.etsi.org/deliver/etsi_tr/102700_102799/102783/01.01.02_60/tr_102783v010102p.pdf.
[6] P. Ferguson and G. Huston. “Quality of service in the internet: Fact, fiction, or compromise?”, Julho 1998. www.potaroo.net/papers/1998-10-qos/qos.ppt. Consulta em 23 de Junho de 2010.
[7] Cisco Visual Networking Index. “Cisco visual networking index: Global mobile data traffic forecast update”, Fevereiro 2010. Disponível em http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html. Acesso em 20 de Junho de 2010.
[8] Wilcox Howard. “Mobile payment markets: Digital & physical goods 2008 – 2013”, Agosto 2008.
[9] Jupiter Research. “Mobile music - videos, streamed, full tracks, ringbacks, ringtones & downturn analysis 2009-2013”, 5ª edição, Maio 2009.
[10]Zambelli Alex. “The birth of smooth streaming”, Fevereiro 2009. Disponível em http://alexzambelli.com/blog/2009/02/04/the-birth-of-smooth-streaming/. Consulta em 21 de Junho de 2010.
[11]Rao A., Lanphier R. “Real time streaming protocol (RTSP)”, Abril 1998. Disponível em http://www.ietf.org/rfc/rfc2326.txt. Consulta em 5 de Dezembro de 2009.
[12]Fecheyr-Lippens Andrew. “A review of HTTP live streaming”, Janeiro 2010. Disponível em http://andrewsblog.org/a_review_of_http_live_streaming.pdf. Consulta em 7 de Março de 2010.
[13]Schulzrinne H., Casner S., Frederick R., Jacobson V. “RTP: A transport protocol for real-time applications”, Julho 2003. Disponível em http://www.ietf.org/rfc/rfc3550.txt. Consulta em 11 de Dezembro de 2009.

[14]McLean P. “Apple launches http live streaming standard”, Julho 2009. Disponível em http://www.appleinsider.com/articles/09/07/08/apple_launches_http_live_streaming_standard_in_iphone_3_0.html. Consulta em 27 de Janeiro de 2010.
[15]Althos. “Mpeg transport stream”. Disponível em http://www.iptvdictionary.com/iptv_dictionary_MPEG_Transport_Stream_TS_definition.html. Acesso em 20 de Fevereiro de 2010.
[16]Apple Inc. “Http live streaming overview”, Setembro 2009. Disponível em http://developer.apple.com/iphone/library/documentation/networkinginternet/conceptual/streamingmediaguide/StreamingMediaGuide.pdf. Consulta em 27 de Janeiro de 2010.
[17]Pantos Ed. “Http live streaming - draft-pantos-httplive-streaming-02”, Outubro 2009. http://tools.ietf.org/html/draft-pantos-http-live-streaming-02. Consulta em 27 de Janeiro de 2010.
[18]Zambelli A. “IIS smooth streaming technical overview”. Technical report, Março 2009. Disponível em http://www.microsoft.com/downloads/details.aspx?FamilyID=03D22583-3ED6-44DA-8464-B1B4B5CA7520&displaylang=en&displaylang=en. Consulta em 23 de Junho de 2010.
[19]FMS GURU. “O que é flash media server?”, Setembro 2009. Disponível em http://fmsguru.com.br/2009/09/15/o-que-e-flash-media-server/. Consulta em 17 de Fevereiro de 2010.
[20]Larson-Kelley L. “Adobe flash media server 3.5 - technical white paper”, 2008. Disponível em http://www.adobe.com/products/flashmediaserver/pdfs/fms3_5_wp_ue.pdf. Consulta em 17 de Fevereiro de 2010.
[21]Adobe Systems Inc. “Actionscript 3.0 native bandwidth detection”. Disponível em http://help.adobe.com/en_US/FlashMediaServer/3.5_Deving/WS5b3ccc516d4fbf351e63e3d11a0773d56e-7ffa.html. Consulta em 15 de Fevereiro de 2010.
[22]Adobe Systems Inc. “RTMP specification”. Disponível em http://www.adobe.com/devnet/rtmp/pdf/rtmp_specification_1.0.pdf. Consulta em 17 de Fevereiro de 2010
[23]Hassoun D. “Dynamic stream switching with flash media server 3”, 2008. Disponível em http://www.adobe.com/devnet/flashmediaserver/articles/dynamic_stream_switching.html. Consulta em 17 de Fevereiro de 2010.
[24]Chriskno. “Adaptive streaming comparison”, Janeiro 2010. Disponível em http://learn.iis.net/page.aspx/792/adaptive-streaming-comparison/. Consulta em 20 de Março de 2010.
[25]Adobe Systems Inc. “Flash player version penetration”, Março 2010. Disponível em http://www.adobe.com/products/player_census/flashplayer/version_penetration.html#ft1. Consulta em 7 de Maio de 2010.
[26]PT Inovação SA. “Windless Streaming Server datasheet”. Setembro 2009. Disponível em http://www.ptinovacao.pt/cms/source/pdf/5f5b8723497236e02653ae9a4e366102.pdf. Consulta em 20 de Outubro de 2009.
[27]Charles M. Kozierok. “OSI reference model layer summary”. Disponível em http://www.tcpipguide.com/free/t_OSIReferenceModelLayerSummary.htm. Consulta em 27 de Maio de 2010.
[28]Charles M. Kozierok. “Transmission Control Protocol (TCP)”. Disponível em http://www.tcpipguide.com/free/t_TCPIPTransmissionControlProtocolTCP.htm. Consulta em 22 de Maio de 2010.

[29]Charles M. Kozierok. “User Datagram Protocol (UDP)”. Disponível em http://www.tcpipguide.com/free/t_TCPIPUserDatagramProtocolUDP.htm. Consulta em 22 de Maio de 2010.
[30]Projeto Kepler. “Lua: Conceitos básicos e API C”, 2008. Disponível em http://www.keplerproject.org/docs/apostila_lua_2008.pdf. Consulta em 18 de Abril de 2010.
[31]lua.org. “A linguagem de programação lua”. Disponível em http://www.lua.org/portugues.html. Consulta em 15 de Abril de 2010.
[32]Gerald Combs. “Wireshark”, 1998. Disponível em http://www.wireshark.org.
[33]ITU. “Methods for subjective determination of transmission quality”, Junho 1998. Disponível em http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-P.800-199608-I!!PDF-E&type=items. Consulta em 22 de junho de 2010.