FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Distribuição de Publicidade Nativa e Contextualizada sobre Ambiente Cloud

Pedro Romano de Oliveira e Silva Barbosa

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Maria Teresa Magalhães da Silva Pinto de Andrade

30 de Julho de 2018

Distribuição de Publicidade Nativa e Contextualizada sobre Ambiente Cloud

Pedro Romano de Oliveira e Silva Barbosa

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Nome do presidente do júri

Arguente: Nome do arguente do júri

Vogal: Nome do vogal do júri

30 de Julho de 2018

Resumo

A presença da publicidade na área da multimédia, principalmente em meios de retenção constante, como é o caso do vídeo e conteúdo *streaming*, tem vindo a aumentar a um ritmo incessante assim como também tem havido esforços no sentido de a anular. A existência de software capaz de remover ou impossibilitar a distribuição de publicidade prova ser um grande obstáculo a quem beneficia desta e tem havido uma necessidade de resolver este problema. Uma das abordagens para solucionar este problema é a publicidade nativa, isto é, a inserção de publicidade no próprio conteúdo transmitido, deixando esta de fazer parte apenas do meta-conteúdo sendo assim bastante mais complicado de a remover. Esta abordagem não só pretende neutralizar software de remoção de publicidade como também abre portas para novas formas de difusão. É possível acrescentar publicidade dinâmica direcionada a diferentes clientes finais, substituindo certas partes do conteúdo para melhor se adequarem a diferentes locais, pessoas, gostos, entre outros. O processo de inserir publicidade no conteúdo é denominado de *Product Placement* e é muito benéfico para as marcas podendo estas usufruir do potencial da publicidade nativa para acrescentar ainda mais valor. O mercado de publicidade dinâmica é um mercado bastante valorizado e em crescimento contudo a distribuição destas soluções tende a ser um processo dispendioso e que requer uma maior supervisão e análise.

Um sistema que ofereça estas funcionalidades é bastante complexo, apresentando um conjunto de desafios não triviais. O objetivo desta dissertação é abordar alguns desses desafios em particular a conversão de formatos. Irá ser feita também uma análise da eficiência da solução e uma comparação entre as várias tecnologias e métodos utilizados.

Espera-se que a aplicação implementada seja capaz de criar os vários *streams* no formato adequado e disponibilizados num serviço de *cloud* de maneira a que a visualização dos conteúdos seja de boa qualidade e preferencialmente com um débito elevado.

Abstract

The presence of multimedia advertising, especially in terms of constant retention, such as video and streaming content, has been steadily increasing and efforts have been made to overturn it. The existence of software capable of removing or preventing the distribution of advertising proves to be a major obstacle to those who benefit from this therefore there has been a need to solve this problem. One of the approaches to solve this problem is native advertising, that is, the insertion of advertising directly in the transmitted content without the need for metadata which removal proves to be much more complicated. This approach not only aims to neutralize advertising removal software but also opens up new ways of distribution. It's possible to add dynamic advertising targeted to different end customers, replacing certain parts of the content to better suit different locations, user characteristics, tastes... The process of inserting advertising into content is called Product Placement and is very beneficial for brands because they can take advantage of the potential of native advertising to add even more value to their advertising campaigns. The dynamic advertising market is a valued and is a growing market, however the distribution of these solutions tends to be a costly process and requires more supervision and analysis.

A system that offers these functionalities is quite complex, presenting a set of nontrivial challenges. The purpose of this dissertation is to address some of these challenges in particular format conversion. An analysis of the efficiency of the solution and a comparison of the various technologies and methods used is also covered in this thesis.

It is expected that the implemented application will be able to create the various streams in the proper format and made available in a cloud service so that the display of the contents is of good quality and preferably with a high throughput.

Agradecimentos

Gostaria de agradecer a todos as pessoas que me apoiaram nesta grande aventura que foi a construção e escrita desta dissertação pois foram indispensáveis em todos os momentos deste processo.

Quero agradecer aos meus pais e à minha família que me proporcionaram excelentes condições durante a toda a minha vida e durante este percurso académico. Sempre me apoiaram, independentemente das situações, e por isso estou eternamente grato.

Aos meus amigos, agradeço o apoio, a paciência e a resiliência em momentos mais difíceis, mas também a sua amizade e o seu apoio que me demonstraram em qualquer etapa da minha vida.

À MOG Technologies por proporcionar um excelente ambiente de trabalho e desenvolvimento que me permitiu evoluir bastante e alcançar os objetivos pretendidos. Um especial agradecimento ao Eng. Pedro Santos que me acompanhou durante todo este processo e me ajudou a procurar novas maneiras de abordar os assuntos mais complexos. Agradeço também a todos os meus colegas de trabalho ,assim como a todos os colaboradores da MOG, que ajudaram a tornar o trabalho desenvolvido ainda mais empolgante e satisfatório.

À FEUP e a todos os envolvidos durante a minha formação académica nesta instituição com especial agradecimento à minha orientadora, Maria Teresa Andrade, que me acompanhou e ajudou durante todos os momentos deste processo.

Pedro Romano Barbosa

"What you aim at determines the way the world manifests itself to you"

Jordan Bernt Peterson

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AAC	Advanced Audio Coding
AES	Advanced Encryption Standard
API	application programming interface
AVC	Advanced Video Coding
AVI	Audio Video Interleave
BBC	British Broadcasting Corporation
CDN	Content Delivery Network
CPU	central Processing Unit
DASH	Dynamic Adaptive Streaming over HTTP
DRM	Digital Rights Management
DVB	Digital Video Broadcasting
ESC	Extração, Sobreposição e Concatenação
FFMPEG	Fast Forward MPEG
HDS	HTTP Dynamic Streaming
HEVC	High Efficiency Video Coding
HLS	HTTP Live Streaming
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IAB	Interactive Advertising Bureau
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IIS	Internet Information Services
ISO	International Organization for Standardization
ISP	Internet Service Provider
K8s	Kubernetes
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MSE	Media Source Extensions
MSS	Microsoft Smooth Streaming
NAL	Network Abstraction Layer
NAT	Network Address Translation
PHP	PHP Hypertext Preprocessor
PPS	Picture Parameter Set
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
RTP	Real-time Transport Protocol
RTPS	Real Time Streaming Protocol
SPS	Sequence Parameter Set

SS	Segmentação e Sobreposição
Symlink	Symbolic Link
TCP	Transmission Control Protocol
TS	MPEG Transport Stream
TV	Television
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VOD	Video On Demand
VAST	Video Ad Serving Template
VMAP	Video Multiple Ad Playlist
VPAID	Video Player Ad-Serving Interface Definition
XML	eXtensible Markup Language
YAML	YAML Ain't Markup Language

Capítulo 1

Introdução

A tecnologia tem vindo a revolucionar inúmeros aspetos da sociedade ao longo dos tempos, mudando a maneira como o ser humano olha para o mundo ao seu redor e como este se comporta e comunica com os outros. De facto, esta é, na atualidade, indispensável para a vida moderna no dia-a-dia de um indivíduo, pois este depende dos seus mecanismos e confia nesta para o auxiliar durante o seu percurso. A revolução da era digital permitiu que fosse possível o armazenamento de informação sem ser necessário um suporte físico e mudou a maneira como a informação é partilhada e percecionada. Os computadores pessoais foram também uma marca muito importante na História pois tornaram possível a muitas pessoas experimentar e usufruir dos múltiplos benefícios dos computadores. Com a popularização da Internet nestas duas últimas décadas a sociedade sofreu uma mudança de paradigma em muitos aspetos. Várias indústrias foram também afetadas por este fenómeno e muitas outras nasceram para dar resposta às novas necessidades. A indústria do entretenimento foi uma das que mais beneficiou com esta alteração e foi responsável por grandes avanços tecnológicos na área de partilha de conteúdo. Foram criadas novas plataformas de distribuição de conteúdo multimédia e as outras existentes modificaram a maneira como as pessoas interagiam com elas.

No sentido de promover os seus produtos e marcas, as empresas aproveitaram esta nova mudança para aumentarem a sua presença e desenvolverem novas maneiras de fazer publicidade. Para alcançarem um equilíbrio entre a maximização da exposição dos seus produtos e a satisfação do cliente final é necessário uma abordagem diferente da tradicional, um tipo de publicidade que vá de encontro às necessidades do utilizador, de caráter mais personalizado e contextual de forma a melhorar as relações entre as duas entidades. O vídeo é um dos veículos multimédia com uma grande potencialidade para os anunciantes pois é o meio que mais consegue cativar o público. A inclusão de publicidade em vídeo é um tema muito presente na comunidade multimédia e tem havido esforços no sentido de desenvolver soluções que vão ao encontro desta necessidade.

1.1 Contexto

O tema desta dissertação foi proposto pela MOG Technologies, uma empresa portuguesa, especializada no setor multimédia há mais de 10 anos cujo objetivo é ser uma das grandes empresas no mercado a nível mundial a oferecer soluções e plataformas que automatizam os processos de trabalho dos operadores de conteúdo multimédia profissional (produtores de vídeo e operadoras de televisão). Enquadrada no ambiente de desenvolvimento de *software*, esta dissertação tem o objetivo de, recorrendo às tecnologias de estado da arte assim como às ferramentas desenvolvidas pela MOG Technologies, especificar, desenvolver e analisar uma nova solução tecnológica. Esta solução irá utilizar aspetos já existentes numa dissertação de um antigo estudante, André Regado, para servir como base de implementação e construir por cima desta, de maneira a concluir o trabalho proposto.

1.2 Motivação e Objetivos

A publicidade tem sofrido, ao longo do tempo, transformações que têm como objetivo adaptar esta a novos meios de comunicação e difusão de informação. Os anunciantes pretendem sempre maximizar a exposição dos seus produtos e marcas de maneira a alcançar o maior número possível de público alvo. No meio de retenção visual e auditiva constante, como é o caso do vídeo, a publicidade é um aspeto que não pode ser alienado quando se fala sobre plataformas de conteúdo multimédia que reproduzem vídeo. Juntamente com o desenvolvimento de novas formas de fazer publicidade nestes meios, ferramentas que permitem ao utilizador eliminar a publicidade indesejada começam a ser bastante utilizadas. É o caso dos *ad blockers* que tem a capacidade de ocultar da vista do utilizador ou até eliminar anúncios existentes em *websites* assim como a publicidade antes, durante e após os vídeos. Esta possibilidade advém da separação do conteúdo com os anúncios exibidos o que torna mais simples, a este tipo de programas, a sua identificação e consequente remoção. A existência deste tipo de *software* e a sua popularização provam ser um grande obstáculo para as empresas que dependem das receitas geradas a partir da publicidade nestes ambientes. O desenvolvimento de soluções capazes de combater este problema é, neste momento, um grande foco de discussão nesta indústria. Uma abordagem a este problema é a publicidade nativa que, como o nome indica, é um tipo de publicidade que é inerente ao conteúdo, isto é, o conteúdo publicitário fica embutido no próprio conteúdo o que torna mais complexa a sua deteção. Este tipo de publicidade, para além deste aspeto, tem outro conjunto de vantagens que são aliciantes ao anunciante.

O objetivo desta dissertação será especificar e desenvolver uma plataforma *cloud* capaz de inserir dinamicamente segmentos publicitários no conteúdo de uma *stream*. Esta publicidade deverá ser contextual e baseada na localização da rede do utilizador final. Este sistema deverá utilizar tecnologia do estado da arte assim como formatos abertos de codificação para selecionar qual o segmento de publicidade mais adequado ao cliente. Para este efeito foram enumerados alguns objetivos chave para este projeto:

- Realizar um levantamento e analisar as diversas tecnologias de *streaming* e codificação de vídeo assim como as principais soluções de inserção dinâmica de anúncios.
- Implementar uma plataforma *cloud* capaz de inserir dinamicamente publicidade contextualizada numa *stream* de vídeo de acordo com a localização do utilizador.
- Criar um conjunto de interfaces abertas que permitam a comunicação entre o servidor dinâmico de conteúdo e os mecanismos de armazenamento destes.
- Analisar e realizar testes ao trabalho realizado de maneira a torna-lo mais eficiente em termos de rapidez e comutação.

1.3 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais quatro capítulos. No segundo e no terceiro capítulo serão respetivamente analisados dois temas, a publicidade e o *streaming* de vídeo de maneira a proporcionar uma visão geral dos temas abordados e das tecnologias estudadas assim como uma comparação entre elas. O capítulo 4 apresentará trabalhos relacionados ao tema que contêm aspetos relevantes para a conceção e implementação da solução. No capítulo 5 será abordada a solução trabalhada e os seus aspetos técnicos assim como os processos e métodos associados. Os testes e discussões estarão inseridos no capítulo 6. No capítulo 7 é feita uma conclusão sobre o desenvolvimento deste trabalho, a satisfação dos objetivos propostos e serão indicados futuros trabalhos.

Capítulo 2

Publicidade

A publicidade já é tão antiga como a própria história das primeiras civilizações. A necessidade de divulgar, anunciar e tornar algo público com o intuito de disseminar informação já tem acompanhado o progresso da humanidade desde a antiguidade. De facto, os egípcios utilizavam o papiro como suporte para os seus cartazes e anúncios e muitos estabelecimentos e entidades recorriam a pinturas nas paredes em pedra para promoverem os seus produtos [ash05]. O uso de gravuras e pinturas para meios de divulgação remonta já os anos 4000 a.C. com a cultura indiana [ash05] e desde então a publicidade tem vindo a evoluir, não só na sua forma de disseminação de informação e anúncio, mas também como um conceito, algo cultural e objeto de estudo, passando a ser vista como algo inerente às civilizações humanas e com um grande valor para os mercados atuais. A publicidade como a conhecemos nos dias de hoje começou a sua era moderna no século XIX onde, motivada pela revolução industrial [AdA03], começaram a ser usadas imagens e frases talhadas com o objetivo de vender produtos e serviços. Foi por volta dessa altura que começaram a fazer parte do vocabulário novos conceitos como por exemplo o de *slogan*. Desde então a sua evolução e transformação está estritamente relacionada com o desenvolvimento da tecnologia. Com a invenção de novos meios de difusão, como por exemplo a televisão e o rádio, a publicidade tem vindo a transformar-se e, de certo modo, reinventar-se no sentido de conseguir adaptar-se a estes e aproveitar o seu poder de disseminação. O mesmo acontece com a chegada da Internet, onde a existência da publicidade é algo que já consideramos como sendo a norma.

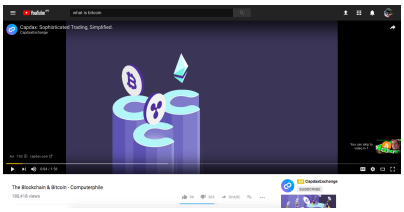
Neste capítulo são expostas as diferentes técnicas e métodos de inserção de publicidade em conteúdo multimédia, que, ao longo do tempo, têm sido desenvolvidos pela indústria, no sentido de identificar as suas características e perceber quais são os desafios ainda existentes para a sua implementação.

2.1 A Publicidade na Atualidade

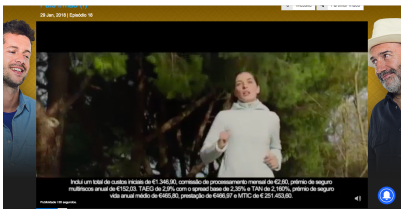
Nos dias que correm a publicidade é cada vez mais presente e tem como intuito a exposição de produtos e serviços de maneira a cativar as pessoas a adquirirem o objeto publicitado. O valor da publicidade e os benefícios que esta traz para o setor comercial são, portanto extremamente elevados, e as empresas interessadas em promover os seus produtos a um público-alvo são capazes de despender grandes quantias em troca da visibilidade dos seus anúncios. São feitos muitos esforços por parte destas entidades no sentido de produzir anúncios que consigam chegar mais eficazmente ao público desejado.

2.2 Publicidade em *Streaming* de Vídeo

Um meio que tem vindo a evoluir no que toca à inserção de conteúdo publicitário é o vídeo pois é neste que existe uma maior liberdade criativa e consegue albergar um grande leque de ferramentas indispensáveis no que toca a criar um anúncio audiovisual (perceção visual, auditiva, entre outros). A publicidade através de vídeo já se encontra bastante presente nos meios de difusão de conteúdo manifestando-se, por exemplo, em anúncios nos intervalos dos programas de televisão, cartazes digitais ou em *streaming* de vídeo em plataformas de conteúdo multimédia (ex: Youtube, Facebook).



(a) Publicidade antes do início de um vídeo na plataforma Youtube



(b) Exemplos de publicidade em plataformas de distribuição de conteúdo multimédia

Figura 2.1: Exemplos de publicidade em plataformas de distribuição de conteúdo multimédia

A forma de fazer publicidade em plataformas de *streaming* de vídeo é idêntica à maneira como a publicidade é abordada na televisão com algumas diferenças. No meio televisivo existe uma quebra nos segmentos do programa para expor a publicidade de um produto. Os anúncios podem ser exibidos no início, a meio ou então no fim do programa televisivo, os chamados *ad breaks* [Reg17].

A IAB, uma entidade que desenvolve standards e especificações para a comunidade multimédia, especifica dois formatos distintos de anúncios para a publicidade nos vídeos digitais:

- Anúncios Lineares: Este tipo de anúncios caracterizam-se por interromperem o seguimento do conteúdo do vídeo e podem ocorrer antes do vídeo (*pre-rolls*), a meio (*mid-rolls*) ou quando o vídeo termina (*post-rolls*). Podem ser interativos e serem complementados por anúncios fora da tela.

- Anúncios Não Lineares: São apresentados como anúncios sobrepostos ao conteúdo do vídeo. Desta maneira não interrompem o seguimento do conteúdo e, idealmente, deverão ser pequenos textos, ícones ou imagens que não obstruam demasiado o conteúdo e por essa razão estão normalmente localizados nas extremidades da tela.

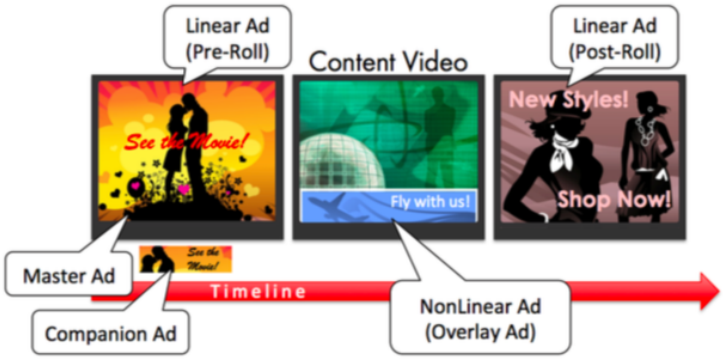


Figura 2.2: Esquema representativo dos vários tipos de anúncios [IAB13]

O método utilizado para os anúncios de televisão não é o suficiente para reter a atenção do consumidor televisivo pois este pode simplesmente ignorar os segmentos publicitários, podendo exercer outras atividades que não a visualização do anúncio [Reg17]. A diferença entre os dois meios é que, no caso do *streaming* de vídeo, o conteúdo é passado *on demand*, ou VOD (*video on demand*), isto é, o utilizador decide quando quer visualizar o conteúdo, podendo até saltar temporalmente, o que não acontece na programação tradicional em que o seu conteúdo é transmitido ao mesmo tempo para todos os utilizadores. No caso da programação linear, o fornecedor de serviços escolhe uma grelha de conteúdo e difunde para todos os telespetadores da mesma maneira, enquanto no serviço a pedido (VOD) o utilizador consegue visualizar o conteúdo em qualquer momento. O progresso tecnológico da televisão não pretende desacelerar e em consequência disso os principais fornecedores de serviço televisivo também têm vindo cada vez mais a adotar VOD na solução televisiva, passando esta a ser transmitida através de *streams* de vídeo digital. Esta solução torna-se mais apelativa ao consumidor pois confere uma maior liberdade de escolha e controlo. Apesar da implementação de sistemas capazes de suportar este tipo de plataformas ser mais custosa e complexa o retorno de investimento prova ser um grande incentivo para as empresas multimédia e operadoras optarem cada vez mais por esta abordagem [Int17].

O facto do conteúdo publicitário ser mantido separado do conteúdo principal leva a que seja mais simples a remoção da publicidade através de aplicações externas. A popularização de *software* capaz de remover e bloquear publicidade assim como a sua consequente popularização tem vindo a provar ser um grave problema no que toca à publicidade digital. Atualmente a maneira mais popular de remover anúncios e publicidade é através de *plug-ins* que podem ser instalados nos *browsers* atuais como é o caso do Adblock Plus [Mor17]. Estas ferramentas permitem filtrar, do conteúdo da página web ou aplicação móvel recebida pelo cliente, as partes relativas à publicidade resultando assim numa página que só contém o conteúdo desejado [Mor17].

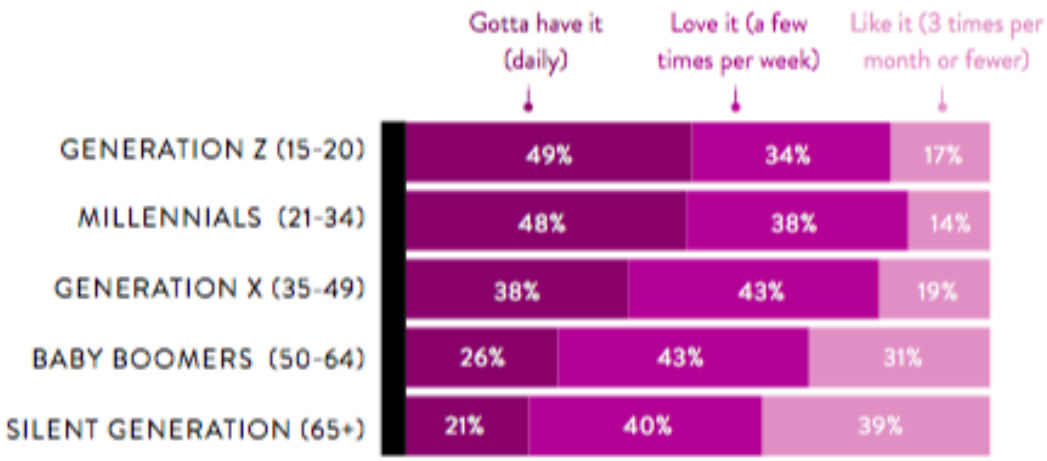


Figura 2.3: Gráfico de comparação da utilização de VOD em camadas etárias diferentes[Nie16]

2.3 Product Placement

Uma técnica utilizada pelas empresas para promover os seus produtos e marcas tem como princípio a inserção de referências visuais no conteúdo de diversos meios tais como os filmes e conteúdo televisivo, isto é, várias referências visuais ou auditivas são usadas com efeito publicitário sem, no entanto, recorrer a publicidade explícita [Reg17]. A utilização de produtos de uma marca existente por uma ou várias personagens em certas cenas de uma longa-metragem, constitui, de facto, um exemplo desta técnica que tem vindo a ser bastante utilizada até aos dias de hoje.

É uma maneira mais subtil de vender um produto que a publicidade tradicional, aumenta a visibilidade da marca e pode fortalecer a opinião do consumidor em relação a esta [AL12]. Esta estratégia de publicidade tem a vantagem, para o espetador, de não ser particularmente intrusiva e não quebra necessariamente o fluxo de visualização do consumidor podendo até fortalecer a experiência deste devido à robustez e ao nível de realismo que o *product placement* pode trazer para este tipo de meios. De facto, a existência de produtos e marcas conhecidas pode ajudar na imersão do espetador, em certos tipos de conteúdos audiovisuais, se o objetivo deste for retratar ao máximo o mundo atual [Reg17];[BKP06].

Contudo, o *product placement* pode ter consequências negativas quando não é bem implementado. Cowley e Barron (2008)[CB08] concluem que, numa situação de colocação proeminente, a memória do observador em relação à marca respetiva pode aumentar, porém, se este for demasiado protuberante pode deixar na pessoa um sentimento de repulsa, pois esta vê a colocação da publicidade em questão como algo forçado e como uma tentativa de persuasão. Esta consequência do excesso da exposição de uma marca ou produto pode ter efeitos negativos tanto para a perceção da marca pelo indivíduo e pode deixar neste más impressões do conteúdo envolvente [CB08]. Isto é, quanto mais proeminente for a colocação de um determinado produto, maior vai ser a suscetibilidade do visualizador se sentir que foi enganado no sentido de obter uma experiência errada à expectada ou alvo de publicidade excessiva, o que resulta na desvalorização do que perceciona e

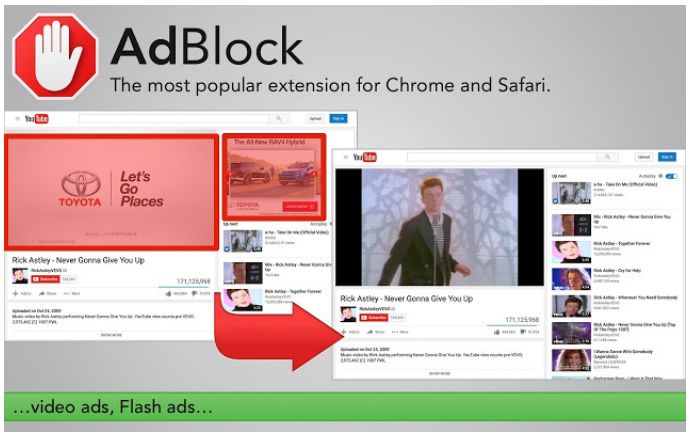


Figura 2.4: Exemplo do resultado da remoção de publicidade recorrendo ao AdBlock

na quebra de imersão que este se encontrava até então.

Segundo Ferraro & Avery[FA00] existem dois tipos de product placement que variam na maneira como são percecionados pelo espetador; proeminentes e subtis. A colocação proeminente de produtos tem como característica, como o nome indica, de ser bastante óbvia para quem está a visualizar e é facilmente detetável. Contrariamente, a colocação subtil passa mais despercebida ao observador e, normalmente, está inserida em segundo plano [FA00].

Para além destes dois tipos de *product placement*, Lehu [Leh07], afirma, relativamente à sua exposição, que existem quatro tipos diferentes:

2.3.1 Colocação Clássica

Este tipo de colocação, tal como o seu nome indica, tem sido praticada desde a introdução do *product placement* e caracteriza-se por expor de maneira natural os produtos como parte da narrativa [AL12]. Uma das desvantagens desta abordagem é que por vezes a publicidade pode passar despercebida aos espetadores devido à grande quantidade de outras colocações de outros produtos existentes na cena e/ou ao longo do conteúdo[AL12]. Cenas cujo cenário é o ambiente citadino é um bom exemplo desta situação devido à grande quantidade de publicidade presente quer seja ela real ou ficcional. Tem a vantagem de ser bastante simples de inserir na cena e tem um custo associado reduzido, e por vezes a sua exposição pode ser gratuita quando aparece sem o consentimento explícito do anunciante [AL12].

2.3.2 Colocação Corporativa

Por vezes é mais simples e vantajoso colocar referências a marcas em vez dos produtos da mesma. Neste tipo de colocação utiliza-se apenas aspetos das marcas familiares ao público como é o caso do logótipo. Para este método ser eficaz é necessário que a marca em questão já possua popularidade suficiente para ser reconhecida pelos espetadores. Se a colocação provar ser eficiente trará efeitos benéficos não só aos produtos da marca respetiva mas também ao sentimento relativo a esta [Leh07].

(a) Filme Terminator 2

(b) Filme Wayne's World

(c) Série de TV Chuck

Figura 2.5: Exemplos de *product placement* em filmes e séries de TV

2.3.3 Colocação Evocativa

Segundo Lehu[Leh07], na colocação evocativa, a abordagem sofre um processo mais discreto no sentido em que o nome do produto assim como o nome ou logótipos da marca não aparecem explicitamente na cena, isto é, o que aparece efetivamente será apenas o produto e/ou a sua embalagem sem quaisquer vestígios relacionados com a marca. É, portanto, necessário que o produto em questão se diferencie do resto dos elementos presentes na cena através de *designs* e embalagens inovadoras e únicas [Leh07]. A desvantagem deste tipo de colocação é o facto de ser necessário conhecer o produto previamente para estabelecer uma ligação à marca, no entanto não é tão intrusivo comparando com os tipos de colocação anteriores.

2.3.4 Colocação Furtiva

A colocação furtiva é o tipo de colocação mais subtil e discreto de entre todos os outros e por vezes é quase impercetível. Por não ser tão visível, quando é detetada provoca um efeito positivo no espetador pois este perceciona a exposição da marca ou produto como algo natural e inerente à cena, ou seja, este considera a colocação do produto ou marca credível no mundo ficcional o que pode proporcionar uma experiência de maior qualidade. [Leh07].

2.4 Publicidade Nativa

A publicidade nativa serve para descrever todo o tipo de publicidade que "toma a forma e aspeto" do conteúdo principal [WE16] e, segundo [WG16], a sua característica principal é a integração dissimulada de maneira a que esta "peça emprestado" a credibilidade do editor. Isto é, em vez dos segmentos de publicidade aparecerem descolados do resto do conteúdo e da forma do produto de multimédia, estes surgem embutidos aparentando fazer parte do corpo do produto. O *product placement* é o precursor da publicidade nativa pois esta beneficia dos vários conceitos presentes na colocação tradicional de produtos mas é mais adequada para conteúdo dinâmico e personalizado. Esta abordagem não só tem o objetivo de melhorar a experiência do utilizador mas também fornecer informação útil e relevante relacionada com o conteúdo visualizado de maneira a manter o utilizador o mais afeiçoado possível, o que se traduz em vantagens para os proprietários do conteúdo principal e para o utilizador, se a informação do anúncio for útil para este [Pul14].

(a) *product placement* proeminente no filme Wayne's World

(b) *product placement* subtil em 007: Quantum of Solace

Figura 2.6: Exemplos de publicidade proeminente e subtil

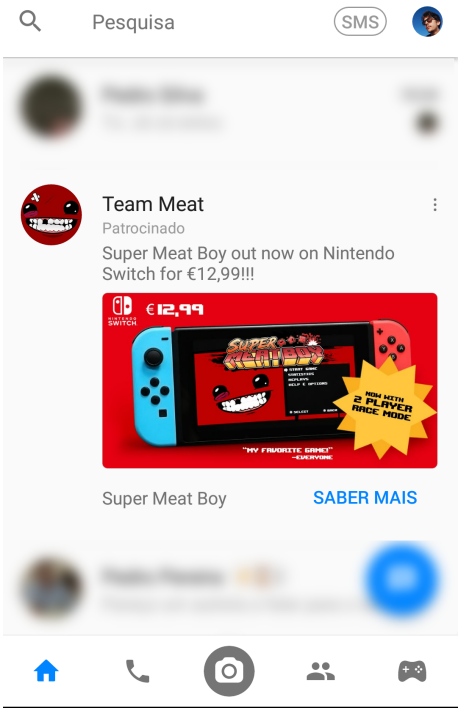


Figura 2.7: Exemplo de publicidade nativa na aplicação Messenger num dispositivo Android

A partir dos resultados obtidos por [TvR12], a publicidade nativa tem o efeito pretendido de mudar a perceção do público em relação à publicidade digital como sendo intrusiva devido à relevância que esta publicidade tem para quem a consome. Consideram também que este tipo de publicidade é adequado e abrange qualquer público-alvo, no entanto é mais eficiente em camadas da população mais jovem devido à fácil penetração em dispositivos, canais de comunicação e plataformas digitais.

Se o objetivo de um anunciante é maximizar o contexto publicitário e se optar, sobretudo pela publicidade digital nativa, este encontrará uma complexidade agravada. Isto porque o contexto da publicidade face ao conteúdo é um fator crucial no que toca à perceção do observador e se esta for feita em grande quantidade ou não se inserir naturalmente no conteúdo pode comprometer a sustentabilidade do modelo de negócio da plataforma [HBH17].

Este tipo de publicidade tem vindo a crescer ao longo dos últimos anos e de acordo com a Enders Analysis (2016)[Ana16], esta poderá representar 52% de toda a publicidade digital em 2020 depois de um crescimento de 156% desde 2015. Segundo um estudo da empresa [Med16], a publicidade nativa cresceu em 74% só no ano de 2016 e a procura por anúncios nativos alcançou valores que excedem o triplo dos valores registados em 2015. Segundo o mesmo estudo, a publicidade nativa continua a crescer a um ritmo constante enquanto a publicidade tradicional tem vindo a decrescer com algumas exceções em áreas de nicho.

2.5 Publicidade Personalizada

Atualmente, muitas empresas são especializadas a recolher informação sobre os utilizadores a partir da Internet e utilizam esses dados para conceder aos anunciantes a habilidade de personalizar a sua publicidade [Tuc14]. A publicidade personalizada, como o nome indica é um tipo estratégia de publicidade praticada pelas empresas de marketing, que tem como objetivo maximizar as receitas da publicidade mas também analisar e avaliar as campanhas publicitárias. Antes da era da Internet era muito difícil perceber se uma campanha de publicidade fora bem-sucedida ou não, pois as empresas de marketing não possuíam dados suficientes para chegarem às conclusões necessárias. [Ora14]. A partir dos dados recolhidos ao utilizador é possível criar anúncios personalizados de maneira a aumentar a relação que um cliente tem com uma empresa ou produto. Segundo um artigo de Harvard Business Review [DP99], existem quatro fases no que toca à estratégia de marketing personalizado:

1. Identificação: É nesta fase que devem ser recolhidos todos os dados e informação sobre os clientes de uma empresa para estimar as suas preferências, costumes e necessidades que têm de ser satisfeitas;

2. Diferenciação: Os clientes de uma empresa nem sempre são um grupo homogéneo com o mesmo valor, preferências ou prioridades. Neste sentido é necessário segmentar o grupo de clientes em aglomerados mais reduzidos de acordo com certos parâmetros;

3. Interação: Nesta fase é necessário saber como alcançar o cliente e saber quais são os canais de comunicação que este prefere. O cliente será mais recetivo e atento a certos mecanismos de publicidade que outros. É importante que as empresas conheçam e comuniquem com os clientes da melhor maneira possível para continuarem a desenvolver a sua relação com este;

4. Personalização: Nesta última fase é essencial que, a partir dos dados recolhidos nas fases anteriores, o produto ou serviço sofra certas alterações de maneira a melhor acomodar as necessidades presentes do cliente individualmente.

2.5.1 Publicidade Dinâmica

A publicidade dinâmica, como o nome indica, refere-se a qualquer tipo de publicidade que pode variar no seu conteúdo, forma e quantidade ao longo do tempo. Por outras palavras, ao

contrário da publicidade tradicional (estática), cujos anúncios publicitários são sempre idênticos independentemente do utilizador, esta pode mudar de acordo com certos aspetos. Os anúncios presentes em algumas páginas da *internet* são um bom exemplo deste tipo de publicidade. Um utilizador que visite um determinado site pode visualizar anúncios personalizados que mudam consoante a sua atividade *online*. Neste sentido são uma evolução da publicidade personalizada tradicional. Outro exemplo são os anúncios em *streams* de vídeo que podem apresentar anúncios diferentes para utilizadores que se encontram em localizações distintas.

2.6 Soluções e Limitações Atuais

Segundo o artigo de Tim Siglin[Sig16] presente na publicação de janeiro/fevereiro da revista Streaming Media, este conclui que o estado da publicidade em *streaming* de vídeo sofreu várias transformações ao longo dos anos e recentemente tem voltado aos processos iniciais. De facto, nos primeiros anos, quando a tecnologia de serviço de publicidade ainda estava em crescimento, esta era introduzida diretamente na stream (*baked*) e esses anúncios eram denominados de *static ads* (anúncios estáticos). Esta solução provou ser insuficiente dado que não é adequada para anúncios personalizados pois na era da Internet em que é possível chegar a virtualmente qualquer público-alvo os anunciantes deixaram de utilizar esta tecnologia e passaram a enviar pedidos HTML que recebiam os segmentos publicitários à parte [Sig16].

Com o evoluir das tecnologias, hoje em dia, já existe *software* capaz de inserir segmentos publicitários nativos em conteúdo multimédia de maneira eficiente e simples para o utilizador final. Normalmente, esta funcionalidade faz parte de um abrangente leque de outras possibilidades existentes em *software* especializado em aumentar as receitas e melhorar os processos de trabalho de empresas ligadas a multimédia e criadoras de conteúdo.

(a) Adobe PrimeTime

(b) Google's DoubleClick

Figura 2.8: Exemplos de software capaz de inserir publicidade nativa

Apesar das soluções acima mencionadas serem capazes de adicionar a publicidade a meio do *stream* de vídeo estas não permitem uma integração completa da publicidade com o conteúdo. Sendo assim não é possível aproveitar os benefícios que a publicidade nativa oferece utilizando estes programas. Neste tipo de *software* recorre-se apenas ao *ad stitching* (denominação utilizada para a inserção dinâmica de publicidade [Bil15]) de segmentos que não estão sobrepostos ao vídeo, idêntico ao que acontece nos anúncios televisivos em que o programa interrompe para dar lugar aos anúncios. Para empresas que pretendem maximizar a sua receita através do uso da publicidade

nativa esta solução insuficiente pois não permite inserir os anúncios sobrepostos ao conteúdo de maneira dinâmica.

O trabalho desenvolvido nesta dissertação tem como objetivo eliminar algumas das limitações existentes atualmente no que toca à inserção de publicidade simultaneamente nativa e dinâmica em *streams* de vídeo. Desta maneira, será possível inserir anúncios que fazem parte do conteúdo da *stream*, e portanto de difícil remoção, juntamente com a possibilidade da sua variação para uma personalização mais eficaz.

Capítulo 3

Streaming de Vídeo

Ao longo destas últimas duas décadas a incessante procura pelo aumento da largura de banda da Internet quer por parte dos utilizadores quer pelo sector comercial, aliada à evolução tecnológica a nível de técnicas de compressão e o desenvolvimento de novos padrões e formatos colocou rapidamente o *streaming* de vídeo como uma das grandes aplicações indispensáveis no dia a dia de quem usufrui frequentemente da Internet. Preferencialmente, a visualização de vídeo deverá ser instantânea e ocorrer mesmo antes de se descarregar todo o conteúdo, em contraste com o que acontecia antes do *streaming*, em que era necessário descarregar o ficheiro de vídeo antes de poder visualizar o seu conteúdo. É devido a esta vantagem que o *streaming* de vídeo tem sido um tema de investigação desde então, por parte da academia e da indústria, e consequentemente tem sofrido grandes evoluções tecnológicas que têm proporcionado ao utilizador final uma melhor experiência de visualização [LWLZ13].

Neste capítulo será apresentada uma breve visão geral sobre o *streaming* de vídeo, os seus processos tradicionais, assim como a sua evolução até ao *streaming* adaptativo. No que respeita o *streaming* adaptativo, são apresentadas algumas das tecnologias mais utilizadas no mercado assim como comparações entre elas. Este capítulo é pertinente para o contexto da solução desenvolvida pois contribui para o maior conhecimento das várias tecnologias utilizadas e ajuda no processo de escolha da tecnologia para a implementação.

3.1 Evolução do *Streaming* de Vídeo

Segundo B. Li et al (2015)[LWLZ13], o desenvolvimento que ocorreu nesta área pode ser dividido em três épocas distintas:

3.1.1 *Streaming* de Vídeo Servidor-Cliente

Durante a década de 1990 a 2000, foram feitos esforços no sentido de criar novos protocolos capazes de especificar a nova tecnologia de *streaming*. Os reprodutores de vídeo da altura foram

incorporando estes protocolos que lhes permitiam receber as *streams* dos servidores espalhados pela Internet. A arquitetura usada é a de servidor-cliente em que um servidor é um processo que fornece um serviço e um cliente é outro processo que efetua um pedido para utilizar esse serviço [TVS07].

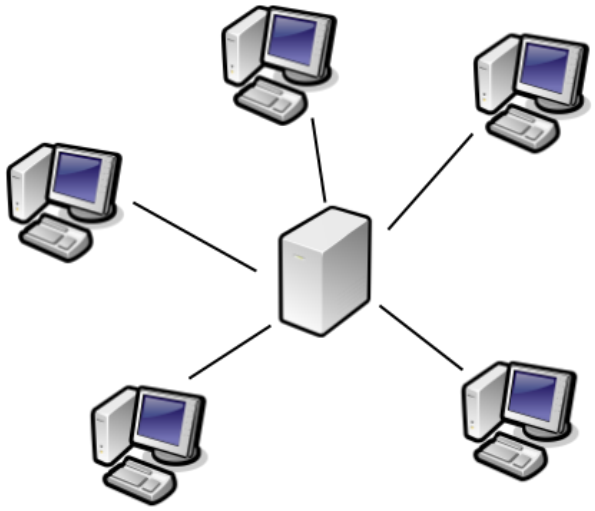


Figura 3.1: Esquema da topologia em estrela

3.1.2 *Streaming* de Vídeo Peer-to-Peer

A necessidade de escalar estes sistemas devido à grande quantidade de utilizadores despoletou, na primeira década de 2000, a investigação e desenvolvimento de técnicas capazes de lidar com este problema. Na arquitetura dos protocolos *Peer-to-Peer* cada cliente também pode ocupar o papel de um servidor, ou seja, é criada uma topologia de rede (descentralizada) ao contrário da arquitetura de cliente-servidor em que a topologia é de estrela (centralizada). Este tipo de sistemas permite uma maior flexibilidade em ambientes em que a escalabilidade é um fator importante assim como a largura de banda.

3.1.3 *Streaming* de vídeo HTTP pela *Cloud*

Para um utilizador poder usufruir dos protocolos P2P(*peer-to-peer*) de *streaming* de vídeo era necessário que este instalasse software que os implementasse. Com a recente popularização do protocolo de HTTP tornou-se mais conveniente para o utilizador consumir *streams* de vídeo através de um web *browser* em vez de descarregar aplicações de terceiros. O protocolo HTTP permite dividir a *stream* de conteúdo em pequenos *chunks* que podem ser sequencialmente descarregados. Esta nova abordagem começa a ser muito utilizada em conjunto com plataformas de computação *cloud*. Segundo um artigo de Tim Siglin[Sig10] de Fevereiro/Março de 2010 para a revista Streaming Media, uma das vantagens de Streaming em HTTP é o cliente poder comunicar de volta com

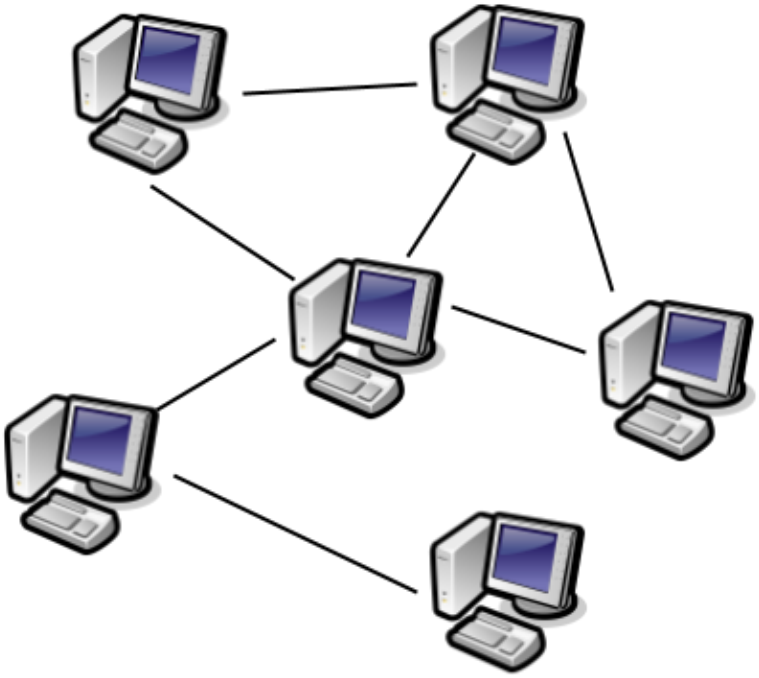


Figura 3.2: Esquema da topologia em rede

o servidor durante o decorrer do vídeo para trocar informações relevantes de maneira a alterar a *stream* de vídeo como, por exemplo, aumentar ou diminuir o débito de transmissão.

3.2 Transmissão de *Streams* de Vídeo

Atualmente, existem duas maneiras de transmitir *streams* de vídeo até ao utilizador final: a transmissão normal ou tradicional e o *streaming* adaptativo.

3.2.1 Transmissão Normal

A transmissão normal utiliza um protocolo denominado de RTP/RTSP. O protocolo RTSP (*Real Time Streaming Protocol*) utiliza as funcionalidades existentes no protocolo de TCP (*Transmission Control Protocol*) para efetuar o controlo da sessão enquanto que o protocolo RTP (*Real-time Transport Protocol*) é utilizado para a transmissão de vídeo e áudio. Neste último são utilizadas as estruturas de transmissão presentes no protocolo UDP que são indexadas e contêm indicadores de prioridade o que permite a ordenação de pacotes fora de ordem e a deteção de erros. Apesar de este protocolo ser ideal em caso de condições ótimas de rede, também possui algumas desvantagens: é pouco eficiente quando as condições de rede estão abaixo do ideal devido ao descarte de pacotes UDP que resulta em artefactos visíveis e faz com que o vídeo tenha pausas e *re-bufferings* frequentes; Por utilizar portas dinâmicas torna os processos de travessia de *firewalls*/NAT mais complicadas podendo utilizar o protocolo TCP ou HTTP como túnel o que

elimina as vantagens da utilização do protocolo UDP; Ao utilizar sessões torna-se difícil e custoso escalar este tipo de sistemas [Swa13]; [Oze11a].

Para além do protocolo RTP/RTSP existe outro protocolo muito utilizado em *websites* de *streaming* de vídeo na Internet. O RTMP, pertencente à empresa Adobe (adquirido à Macromedia), utiliza um único canal TCP para transmitir os dados e, apesar da sua grande adoção na indústria, sofre do mesmo problema que o anterior pois não é adequado para milhões de utilizadores [Reg17]; [Swa13].

3.2.2 Transmissão Adaptativa

Durante os últimos anos, o protocolo HTTP tem vindo cada vez mais a ser utilizado na Internet para distribuir conteúdo multimédia e as soluções existentes implementam um tipo de transmissão denominado de *streaming* adaptativo que permite ao cliente que recebe uma *stream* de vídeo escolher de entre várias opções de compressão aquela que melhor se adequa às condições da sua rede, ao mesmo tempo que o conteúdo é visualizado. Se a rede onde o cliente está inserido sofrer de congestionamento e a sua largura de banda diminuir, o que esta técnica faz é pedir ao servidor segmentos do conteúdo (*chunks*) que estejam codificados com um bit rate menor, de maneira a manter a experiência do utilizador o mais suave possível, mantendo um equilíbrio entre qualidade e velocidade. Esta abordagem beneficia de uma infraestrutura já existente, o protocolo HTTP, que simplifica a implementação desta técnica de transmissão pois esta aproveita as várias características do protocolo. [TG12]; [Reg17]].

Este tipo de transmissão de dados tem os seguintes benefícios relativamente à transmissão tradicional:

- Por utilizar TCP, e particularmente HTTP sobre TCP, atravessar *firewalls* e NATs torna-se mais simples devido aos mecanismos presentes nestes protocolos que já se encontram preparados para funcionar neste tipo de situações [ABD11]. Normalmente, as *firewalls* já se encontram configuradas para suportar conexões de HTTP [Vet11].

- O protocolo HTTP permite controlar as sessões de *streaming* com o servidor e toda a lógica é efetuada no cliente. ou seja, basta os servidores suportarem o protocolo HTTP, como se se tratasse apenas de um servidor Web base. Consequentemente, esta característica oferece a possibilidade de não ser necessário a alocação de recursos adicionais para o servidor ser capaz de fornecer um grande número de clientes [Vet11]; [Sto11].

- Ao utilizar o protocolo HTTP, este tipo de transmissão consegue tirar partido dos mecanismos de *caching* presentes nas redes de ISPs, corporações e organizações o que melhora a qualidade de serviço e a eficiência geral do sistema [Oze11a].

- A lógica a nível do cliente permite-lhe controlar o débito de transmissão (bit rate), sem ter de negociar com o servidor. a partir das características da rede e do ambiente onde este está inserido, como, por exemplo, o poder de processamento do cliente (CPU), a banda

larga disponível, a resolução do conteúdo recebido, o tipo de *codec* utilizado entre outros [Reg17]; [Sto11].

3.2.3 Compressão de Vídeo

A transmissão de conteúdo multimédia com alta qualidade pode tornar-se custosa devido às diferentes características da *stream* transmitida, como por exemplo, a sua resolução, o número de *frames* por segundo, o espaço de cores, entre outros. No sentido de otimizar a eficiência de transmissão, tentando ao mesmo tempo, preservar a qualidade aparente do conteúdo, é utilizada uma técnica chamada de compressão.

A compressão de vídeo e áudio é um processo de transformação que é aplicada num determinado ficheiro ou *stream* de multimédia que reduz consideravelmente o seu tamanho e simultaneamente tenta preservar a sua qualidade. O conteúdo é comprimido utilizando um *encoder*, é transmitido pelos canais respetivos e depois reconstituído recorrendo a um *decoder*. Aos programas utilizados para este processo (compressão/descompressão) são designados de *codecs* (*coder/encoder*).

Existem dois tipos de compressão de vídeo: a compressão sem perdas (*lossless*) que, como o nome indica, reduzem o tamanho do conteúdo sem alterar a qualidade final, e a compressão com perdas (*lossy*) que utiliza várias técnicas de compressão para construir um resultado mais reduzido ainda mas com uma qualidade inferior ao original. Estas técnicas de compressão podem ser categorizadas em dois tipos:

- *Intra-frame*: Esta técnica de compressão é aplicada em cada *frame* do vídeo como se tratasse da compressão de imagem. Para cada *frame* existe um mecanismo de compressão e não existe qualquer relação entre os *frames*.

- *Inter-frame*: A compressão *inter-frame* tira partido da redundância entre os *frames* de vídeo. Esta técnica calcula e armazena apenas as mudanças entre os *frames*. Esta técnica é bastante mais eficiente que a anterior na maioria dos casos.

3.2.3.1 H.264

O H.264, também conhecido por MPEG-4 Part 10, Advanced Video Coding, ou AVC, é um *codec* de vídeo muito utilizado em formatos como o MP4 e o QuickTime (.MOV) para transmissão de *streams* vídeo através da internet. Este padrão tem sido implementado por muitas entidades com diferentes perfis de qualidade e eficiência.

Este *codec* utiliza ambas as técnicas de compressão mencionadas anteriormente (*intra-frame* e *inter-frame*) para comprimir o conteúdo multimédia. Para isso utiliza três tipos de frames:

- *I-Frame*: Este *frame* pode ser considerado como uma imagem completa, pois é comprimido utilizando apenas a técnica de *intra-frame*. A sua construção não depende de mais nenhum *frame* e normalmente são criados quando existe uma diferença brusca de imagens entre os *frames* originais (mudança de cenas ou perspetivas).

- *P-Frame*: Estes *frames* guardam apenas as alterações entre o *frame* atual e o anterior. Por exemplo, no caso de uma cena com um fundo estático, este *frame* apenas armazena a diferença nos pixéis que mudam, guardando assim muito espaço.
- *B-Frame*: Os *frames* B são idênticos aos *frames* P, mas também utilizam os *frames* seguintes para armazenar a diferença.

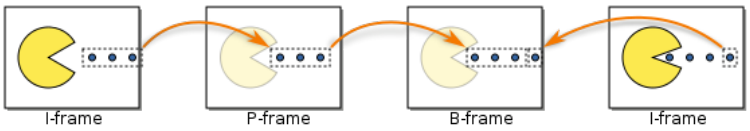


Figura 3.3: Exemplo dos vários tipos de *frames*

3.3 Tecnologias Adaptativas de *Streaming*

O aumento do consumo de dados baseados na Web particularmente a nível de tráfego de vídeo provoca constantemente uma necessidade de aumentar os requisitos de banda larga, e considerando que a Internet não é um sistema com o base na qualidade de serviço (QoS) é necessário desenvolver novas técnicas que se adaptem a este ritmo de consumo e qualidade que os utilizadores esperam comparativamente às exigências presentes em redes de televisão tradicional. Os esforços feitos para atingir estes padrões fizeram surgir um conjunto de novos protocolos desenvolvidos pelas empresas Apple, Microsoft e Adobe, respetivamente, Apple's HTTP Live Streaming (HLS), Microsoft Silverlight Smooth Streaming (MSS), e Adobe's HTTP Dynamic Streaming (HDS) [Net]. Mais tarde, a ISO/IEC (*International Organization for Standardization/International Electrotechnical Commission*) publicou um standard de um novo protocolo não proprietário de distribuição de vídeo denominado de MPEG-DASH que provaria ser um dos protocolos desse tipo mais populares [Fuj].

As tecnologias adaptativas de *streaming* existentes partilham das mesmas características a nível geral da arquitetura, no entanto cada uma apresenta certas diferenças em relação às outras. Estas técnicas separam o ficheiro de vídeo em vários segmentos de diferentes qualidades (armazenados no servidor HTTP) que posteriormente podem ser recuperados a pedido do cliente. Este tem conhecimento destes segmentos a partir de um ficheiro de meta-dados a que se intitula de MPD (*Media Presentation Description*). Este manifesto em XML é responsável por conter referências aos fragmentos assim como os seus atributos respetivos. Desta maneira a lógica fica inteiramente da parte do cliente que escolhe entre as várias qualidades de vídeo. As tecnologias de *streaming* adaptativo acima referidas definem o seu próprio formato do ficheiro de manifesto. [Reg17].

3.3.1 Apple's HTTP Live Streaming (HLS)

O Apple's HTTP Live Streaming é um protocolo proprietário de *streaming* adaptativo HTTP criado pela Apple cuja funcionalidade é distribuir conteúdo multimédia entre dispositivos que

possuam sistemas operativos da Apple (iOS, macOS, Apple TV, ...)[Oze11b]. O HLS permite transmitir vídeo e áudio entre iPhone, iPod Touch, iPad ou Apple TV, transmitir eventos ao vivo sem haver necessidade de servidores especializados e enviar vídeo *on demand* encriptado e com autenticação que permite aos produtores de conteúdo proteger os seus produtos. [Appb].

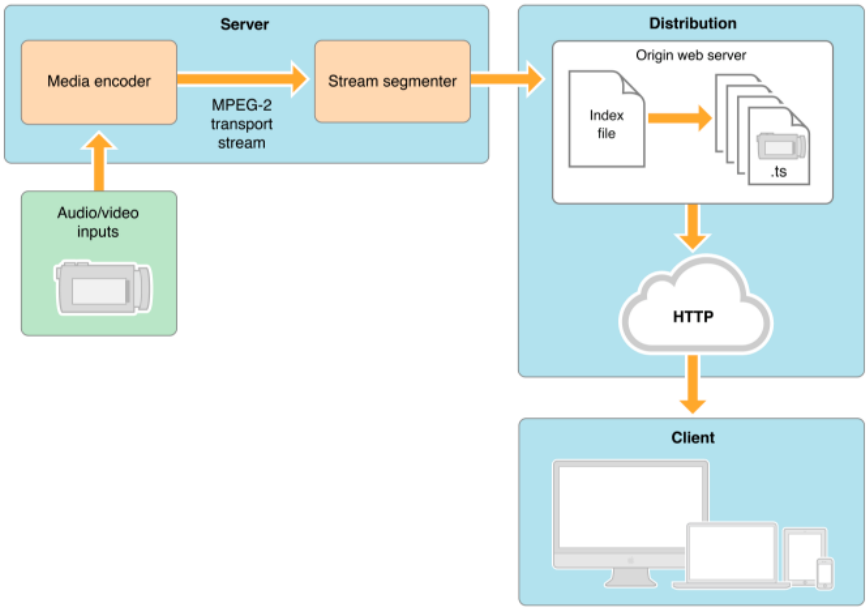


Figura 3.4: Esquema representativo da arquitetura do protocolo HLS [Appb]

Os ficheiros de vídeo devem ser codificados utilizando o codec H.264 para o vídeo e o *codec* de áudio AAC e os *chunks* gerados deverão ser encapsulados numa *stream* de transporte MPEG-2.

Para a preparar uma distribuição com o HLS, é necessário codificar o ficheiro de origem em múltiplos ficheiros de diferentes taxas de informação, normalmente separados em três qualidades (*low, mid, high*). Cada um destes ficheiros deve ser dividido em pequenos segmentos pelo segmentador, com uma extensão .ts, de curta duração (entre cinco a dez segundos) e devem ser armazenados num servidor HTTP em conjunto com um ficheiro geral de manifesto, com a extensão .M3U8, que redireciona o cliente a outros vários ficheiros de manifesto correspondentes a cada *stream* codificada. Os ficheiros de manifesto especificam o perfil da *stream* correspondente o que permite que o player selecione apenas as *streams* compatíveis de acordo com o tipo de dispositivo.

Para o processo após a codificação das *streams* a Apple fornece um conjunto de ferramentas para ajudar na criação dos vários ficheiros necessários à distribuição, entre quais os seguintes:

- O *Media Stream Segmenter* e o *Media File Segmenter* produzem ambos os ficheiros de *chunks* .ts e os ficheiros de índice. O primeiro recebe como entrada uma *stream* de transporte MPEG-2 e o segundo recebe ficheiros H.264

- O *Variant/Master Playlist Creator* reúne todos os ficheiros de índice gerados pelos segmentadores para criar o ficheiro mestre .M3U8 que especifica as diferentes *streams*;

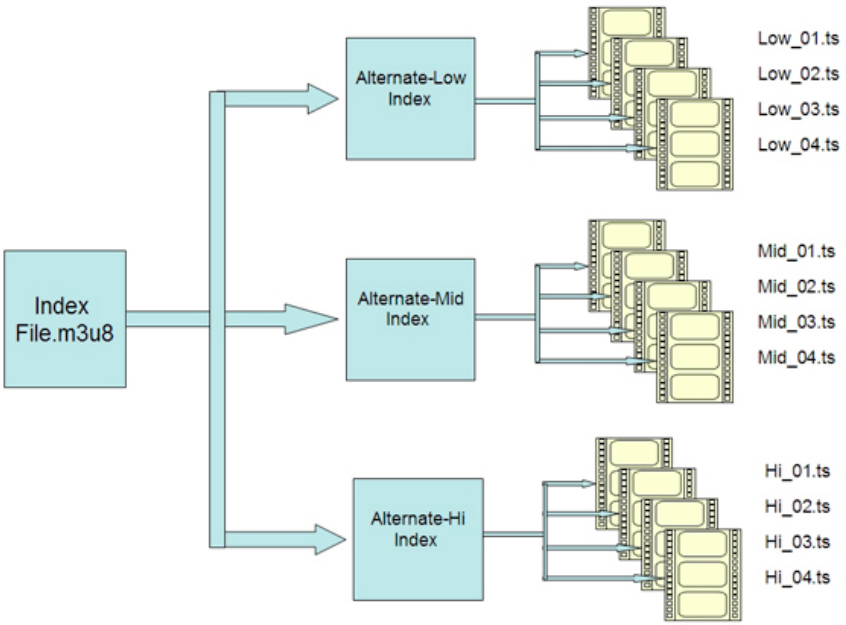


Figura 3.5: Esquema da estrutura de ficheiros do HLS

- O *Media Stream Validator* inspeciona os ficheiros criados para verificar se estes são compatíveis com o protocolo HLS.

A nível da reprodução por parte do cliente, os dispositivos utilizam o web *browser* Safari para correr o vídeo. A Apple recomenda que seja utilizada a *tag* HTML5 <video> para implementação numa página web [Appb].

Existem dois tipos de ficheiros de manifestos com a extensão .M3U8 especificados pelo HLS responsáveis pela indexação dos elementos existentes na arquitetura [Reg17]; [Fis];[Appa]:

- *Normal Playlist*: Este ficheiro contém as referências para os ficheiros de *chunks* .ts e pode ser de dois tipos: *live* e de eventos que por ser usado para *streams* em tempo real requerem que seja sempre efetuadas atualizações aquando de novos segmentos no servidor; VOD (*video on demand*) que contém a lista completa do início ao fim (indicado pela etiqueta "EXT-X-ENDLIST") de todas as referências para os segmentos o que torna possível o *player* navegar por todo a lista e escolher certos instantes.

- *Master Playlist*: Este ficheiro contém uma lista de referências para ficheiros normais de *playlist* acima referidos podendo estes ser caminhos locais ou URLs. Este ficheiro contém outros detalhes importantes utilizados pelo cliente, como a largura de banda, a resolução, e *codecs* utilizados, para este, no momento de decisão, escolher o que melhor se adequa à largura de banda disponível.

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXTINF:9.009,
http://media.example.com/first.ts
#EXTINF:9.009,
http://media.example.com/second.ts
#EXTINF:3.003,
http://media.example.com/third.ts
#EXT-X-ENDLIST
```

(a) Ficheiro *playlist* normal VOD

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:2680

#EXTINF:7.975,
https://priv.example.com/fileSequence2680.ts
#EXTINF:7.941,
https://priv.example.com/fileSequence2681.ts
#EXTINF:7.975,
https://priv.example.com/fileSequence2682.ts
```

(b) Ficheiro *playlist* normal *live*

Figura 3.6: Exemplo de ficheiros *playlists* normais

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=1280000,AVERAGE-BANDWIDTH=1000000
http://example.com/low.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2560000,AVERAGE-BANDWIDTH=2000000
http://example.com/mid.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=7680000,AVERAGE-BANDWIDTH=6000000
http://example.com/hi.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=65000,CODECS="mp4a.40.5"
http://example.com/audio-only.m3u8
```

Figura 3.7: Exemplo de uma *Master Playlist*

3.3.2 Microsoft's Silverlight Smooth Streaming (MSS)

O Microsoft Smooth Streaming é o protocolo desenvolvido pela Microsoft em resposta ao *streaming* adaptativo de maneira a proporcionar uma experiência de multimédia capaz de se adaptar às condições do ambiente onde o utilizador se insere e às capacidades do dispositivo [Micb].

Geralmente, o conteúdo codificado (ISO MPEG-4) é inserido num servidor IIS (*Internet Information Services*) da Microsoft que é responsável pela combinação do vídeo e aúdio em dois ficheiros diferentes, respetivamente do tipo .ismv e .isma, para cada perfil de qualidade [Fis].

O ficheiro de manifesto pode ser adquirido através de um servidor IIS da Microsoft o que não implica que não seja possível adquirir por meio de um servidor HTTP normal. O manifesto recebido pelo cliente é um ficheiro de XML com a extensão .ismc e permite a este realizar um pedido RESTful com a informação temporal de uma *stream*. Este comportamento é radicalmente diferente do protocolo HSL, que divide as *playlists* em *urls* explícitos de cada *chunk* para serem chamados pelo *player*, pois utiliza *timestamps* nos pedidos *url*. As etiquetas denominadas Stream Index especificam uma *stream* que contém informações relativas aos vários perfis e às características da mesma (tipo, níveis de qualidade, tamanhos, entre outros). Dentro de cada destas etiquetas encontram-se outras, entre as quais, as mais relevantes [Mica]:;[Fis]:

1. *QualityLevel*: este elemento é obrigatório para conteúdo de vídeo/áudio e defina um nível de qualidade. Especifica o *bit rate*, a resolução e o *codec*, entre outros;

2. c: elemento que define um *chunk* e descreve os diferentes fragmentos da *stream*. A este está associado um atributo que representa um *timestamp*. Este dado temporal é utilizado no pedido para recolher os fragmentos pretendidos.

Existem vários benefícios desta tecnologia em comparação com as outras, como por exemplo, o facto de o MSS utilizar um número muito reduzido de ficheiros onde agrega toda a informação da *stream* e suportar nos seus ficheiros, múltiplas faixas de informação onde podem ser inseridos meta-dados, legendas, e até segmentos relativos à publicidade. Porém, o MSS adiciona mais uma camada de lógica, com o uso de servidores IIS, o que aumenta a propensão a erros e ao aumento da complexidade [Fis]. Requer também que o *plug-in* da Microsoft Silverlight esteja instalado para poder ser utilizado.

3.3.3 Adobe HTTP Dynamic Streaming (HDS)

O Adobe HTTP Dynamic Streaming é um protocolo proprietário de *streaming* criado pela Adobe e foi desenvolvido para partilhar conteúdo multimédia de maneira eficiente sem ser necessário recorrer a servidores de *streaming* [Ado]. Este protocolo foi definido depois do HLS e do MSS e procura juntar várias características destes.

Na fase inicial o cliente recebe um ficheiro XML com a extensão .f4m que é, tal como o HLS, um ficheiro manifesto com informação dos vários perfis contendo os URLs associados aos segmentos análogo à *Master Playlist* do HLS. Estes dados encontram-se no formato binário e denominam-se de *bootstrap information*. O áudio e vídeo dos segmentos são codificados em ficheiros MP4 tal como é feito no MSS.

"http://server_and_path/QualityModifierSeg'segment_number'–Frag'fragment_number'"

O *url* acima referido é um exemplo de um pedido existente no ficheiro manifesto para um *chunk*. Como se pode verificar, é necessário fornecer ao servidor através do pedido um número de segmento assim como um número de um fragmento para especificar um determinado *chunk* [Fis]. Um fragmento é uma unidade que contém uma determinada quantidade de conteúdo, suficientemente pequena para ser descarregada mas grande o suficiente para ser adequadamente tratada como um objeto na infraestrutura HTTP. É o equivalente a um ficheiro .ts no protocolo HLS, no entanto todos os fragmentos de um determinado segmento estão contidos num único ficheiro com o formato .f4f [Ado].

A arquitetura do protocolo HDS encontra-se dividida em quatro fases distintas [Encc]:

1. Preparação: Nesta fase são criados os ficheiros de meta-dados e de conteúdo (protegido ou não) a partir de recursos VOD ou *live*;

2. Distribuição: Os ficheiros criados são armazenados num servidor HTTP que serve como o espaço de armazenamento de origem. Adicionalmente podem ser utilizados CDNs para tornar o sistema mais eficiente;

3. Proteção: Nesta fase os dados serão protegidos recorrendo a um servidor licenciado pela Adobe através do Adobe Flash Access, utilizado na solução DRM desta empresa desenvolvida para salvaguardar os direitos de autor;

4. Consumo: O consumo do conteúdo pode ser feito através do Flash Player ou a partir de *players* de terceiros.

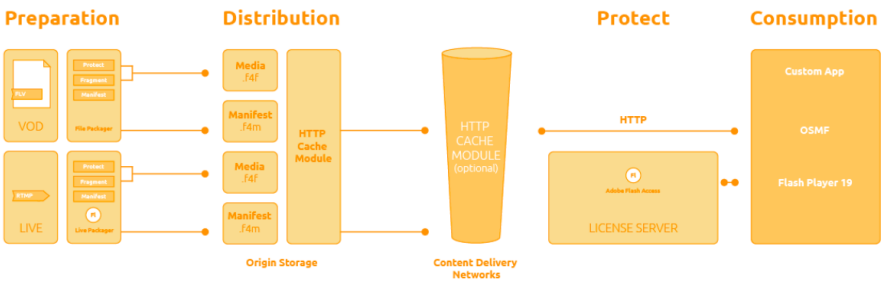


Figura 3.8: Esquema representativo da arquitetura do HDS [Encc]

Por integrar a plataforma Flash, o protocolo HDS beneficia do ambiente deste e da quantidade já existente de profissionais experientes que estão familiarizados com o software. Para além disso o cliente Flash já se encontra instalado em quase todos os computadores pessoais existentes e pode ser instalado numa grande variedade de dispositivos [Fis].

3.3.4 Dynamic Adaptive HTTP Streaming (MPEG-DASH)

A situação fragmentada do mercado antes da criação do MPEG-DASH e as exigências da indústria face às novas tecnologias de *streaming* levaram a que a organização MPEG propusesse, em conjunto com outras instituições de padrões e expertos na matéria, a criação de apenas um único padrão que pudesse ser implementado em diferentes plataformas, *open source* com uma complexidade reduzida e que fosse capaz de responder às necessidades da época [Vet11]. Nessa altura, para um dispositivo conseguir receber conteúdo de cada servidor era necessário que o dispositivo suportasse o protocolo de *streaming* respetivo e era necessário instalar software extra para tal [Encd]. Assim, em 2011, o protocolo MPEG-DASH foi especificado e desde então tem sido bem recebido por toda a comunidade e tem sido adotado por serviços de TV e plataformas multimédia como o iPlayer da BBC, o Youtube e a Netflix [Fuj];

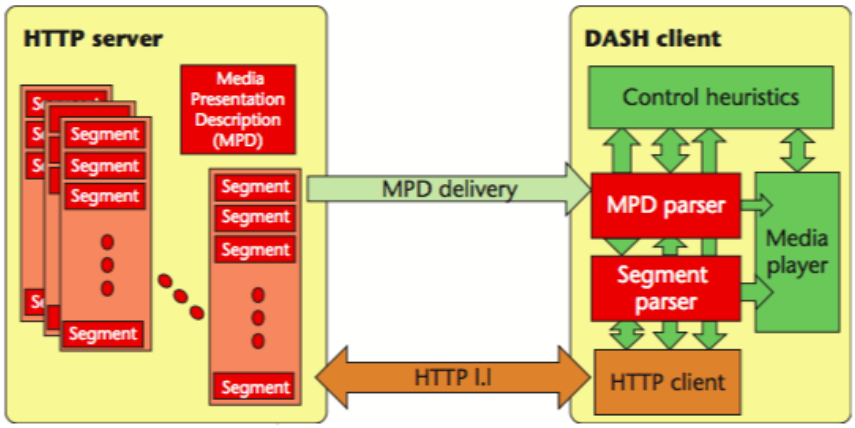


Figura 3.9: Esquema representativo da arquitetura do MPEG-DASH [Vet11]

Como é comum aos outros protocolos, o MPEG-DASH guarda o conteúdo no servidor HTTP separado em conjuntos de segmentos com diversas qualidades e um ficheiro manifesto do tipo XML, tal como o HDS e o MSS, chamado Media Presentation Description (MPD) que guarda as referências aos segmentos em forma de URL. Este ficheiro contém também dados referentes a certas características das *streams* como a disponibilidade do conteúdo, tipos de média, resoluções, limites mínimos e máximos da largura de banda, proteção de direitos digitais (DRM), entre outras. A partir desta informação o protocolo efetua os pedidos necessários para obter os segmentos recorrendo a um pedido HTTP GET. A especificação do MPEG-DASH não refere quais os formatos de conteúdo a serem codificados em segmentos nem a maneira como são efetuadas as heurísticas de adaptação ou o funcionamento do *player* [Vet11]. Sendo assim, poderão existir várias implementações destes vários componentes o que torna o MPEG-DASH o protocolo mais versátil e adaptável, no entanto é o protocolo mais complexo de entre todos os outros.

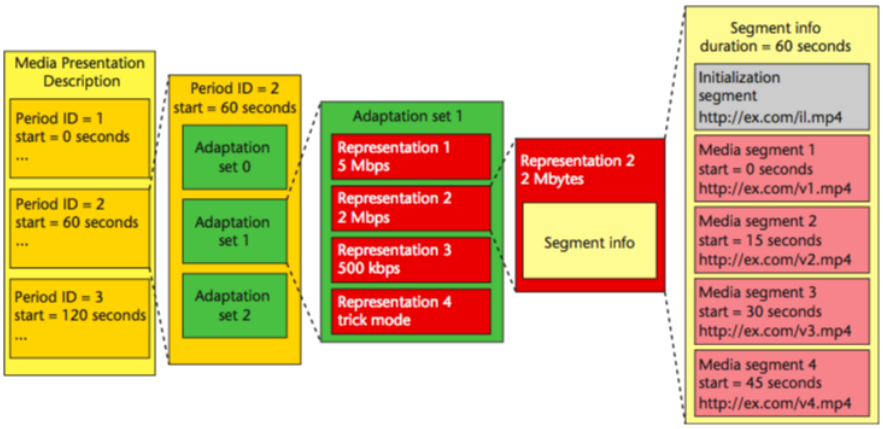


Figura 3.10: Estrutura do ficheiro MPD do MPEG-DASH [Vet11]

O ficheiro MPD constitui o componente central deste protocolo pois é onde está armazenada

a informação que permite este funcionar devidamente. Este ficheiro contém alguns elementos relevantes [Vet11];[Fuj];[Reg17]:

- *Media Presentation Description*: Este elemento é a raiz de um ficheiro MPD e contém metadados relevantes ao programa como, por exemplo, a localização do servidor, a duração total, a duração mínima do *buffer*, entre outros. Um elemento MPD pode conter vários Períodos que são segmentos de uma *stream* divididos temporalmente. Isto permite uma inserção mais simples de conteúdo extra no meio de um vídeo o que é vantajoso para a colocação de publicidade;
- Período: Um período é um elemento que especifica uma parte de um conteúdo de uma *stream*. Contém vários conjuntos de adaptação e é-lhe associado um tempo de início;
- Conjunto de Adaptação: É nestes elementos que estão referenciadas as diferentes representações dos perfis que são escolhidos pelo cliente através de certas heurísticas diferenciados pelo bit rate, resolução e *codecs*;
- Representação: Uma representação é responsável por conter um conjunto de vários segmentos dentro de um elemento denominado Segment Info. este elemento tem de ter um segmento de inicialização antes do resto dos segmentos de vídeo que contém informação relativamente à representação acima na hierarquia.
- Segmentos: Os segmentos são compostos por URLs que representam os *chunks* de conteúdo que o cliente terá de descarregar para reproduzir no *player*.

Existem algumas particularidades, positivas e negativas, relativas a este protocolo que devem ser mencionadas. As vantagens do MPEG-DASH é ser um protocolo aberto, não proprietário, o que facilita a sua adoção em muitos dispositivos e ter um apoio de várias entidades da indústria desde a sua conceção o que resulta em benefícios para a interoperabilidade. No entanto, no caso do MPEG-DASH, sendo um protocolo aberto, as suas diversas implementações poderão não ser compatíveis umas com as outras [Fis].

3.3.5 Comparação das Tecnologias

Todas as tecnologias referidas conseguem alcançar os objetivos que foram propostos para o seu desenvolvimento e comportam-se em adequadamente em ambientes dinâmicos a partir de princípios muito idênticos, porém não só existem grandes diferenças a nível técnico entre elas como também a nível de funcionalidades. Para a seleção de uma tecnologia capaz de responder às necessidades de um projeto que envolva *streaming* adaptativo é necessário recolher e comparar as várias características das diferentes implementações.

3.3.5.1 Comparação de Funcionalidades

- Servidores Comuns de HTTP: Tanto o HLS como o MPEG-DASH podem utilizar servidores de HTTP comuns como, por exemplo, o servidor Apache sem ser adicionada qualquer

	DASH	**HLS**	**MSS**	**HDS**
Servidores comuns de HTTP	+	+	-	-
Múltiplos canais de áudio	+	+	+	+-
Encriptação	+	+	+	+
Legendas/ Legendas Ocultas	+	+	+	+
Inserção de anúncios	+	+-	+-	+-
Troca rápida de canal	+	-	+	+
CDN em paralelo	+	-	-	-
Agnóstico a codecs de vídeo e aúdio	+	-	-	-
Suporte a HTML5	+	-	-	-
Remover/adicionar níveis de qualidade dinamicamente	+	-	-	-

Tabela 3.1: Comparação das funcionalidades de cada tecnologia

camada de lógica do lado do servidor, o que acontece com os protocolos da Microsoft e da Adobe.

- Múltiplos canais de áudio: Esta funcionalidade é importante quando é necessário haver suporte para conteúdo multi-linguístico. MPEG-DASH, HLS e MSS armazenam e entregam o vídeo e áudio separadamente enquanto que o HDS combina os dois canais num fragmento.

- Encriptação: Todas as tecnologias de *streaming* adaptativo aqui referidas suportam a encriptação do conteúdo transmitido. O HLS encripta os ficheiros .ts com a localização das chaves no ficheiro da *playlist*. O MSS utiliza uma plataforma denominada PlayReady que fornece ferramentas de encriptação para gerar chaves e enviá-las ao cliente. O HDS, numa abordagem um pouco diferente dos anteriores, utiliza a plataforma Adobe Access em que a API de encriptação tem de ser implementada pelos vendedores de DRM. O acesso às chaves é feito em tempo de execução. O MPEG-DASH baseia-se num formato comum de encriptação que permite a gestão de conteúdo por parte de outros sistemas de DRM, o que quebra o paradigma de o cliente ter de utilizar um tipo de *player* associado a um tipo de encriptação.

- Legendas e Legendas ocultas: Todos os protocolos referidos suportam legendas que se encontram num ficheiro à parte e são referidas no ficheiro manifesto. No entanto, estes formatos têm alguma dificuldade em suportar legendas de imagens DVB.

- Inserção eficiente de anúncios: De todos, o protocolo MPEG-DASH é o único que especifica uma estrutura, denominada de Período, que permite dividir o e inserir conteúdo extra de maneira simples. Esta característica é vantajosa para a inserção de anúncios pois não é necessária qualquer infraestrutura lógica adicional. O resto dos protocolos consegue inserir anúncios aproveitando o mecanismo de *chunks*.

- Troca rápida de canal: O protocolo HLS divide o conteúdo em *chunks* de 10 segundos cada, o que dificulta a troca rápida de canais pois esta está relacionada com o tamanho dos segmentos. O resto das tecnologias utiliza durações mais pequenas que facilitam este processo.

- CDN em paralelo: O MPEG-DASH é o único protocolo que consegue oferecer ao cliente uma maneira de escolher entre múltiplos CDNs aquele que lhe fornece ligações mais eficientes.

- Agnóstico a codecs: O MPEG-DASH é agnóstico a *codecs* de vídeo e áudio, o que significa que consegue suportar H.264/AVC, H.265/HEVC, MPEG-2 Vídeo, VP8, VP9, entre outros, para vídeo e MP3 e AAC para áudio.

- Suporte a HTML5: O MPEG-DASH pode utilizar as extensões do HTML5, Media Source Extensions (MSE), para reprodução nativa em *browsers*.

- Remover e adicionar níveis de qualidade dinamicamente: Com o MPEG-DASH é possível, durante a *stream* adicionar ou remover perfis de qualidade dinamicamente, através dos Períodos.

Conclui-se que o protocolo capaz de suportar mais funcionalidades de maneira mais eficiente entre todos os outros é o MPEG-DASH, no entanto este é o protocolo mais recente e nem todos os codificadores ou reprodutores o implementam. Apesar do MPEG-DASH oferecer mais funcionalidades, a escolha de protocolo fica resumida à seleção de certas características cruciais aos sistemas que se querem desenvolver [Mue15];[Reg17];[Fis]].

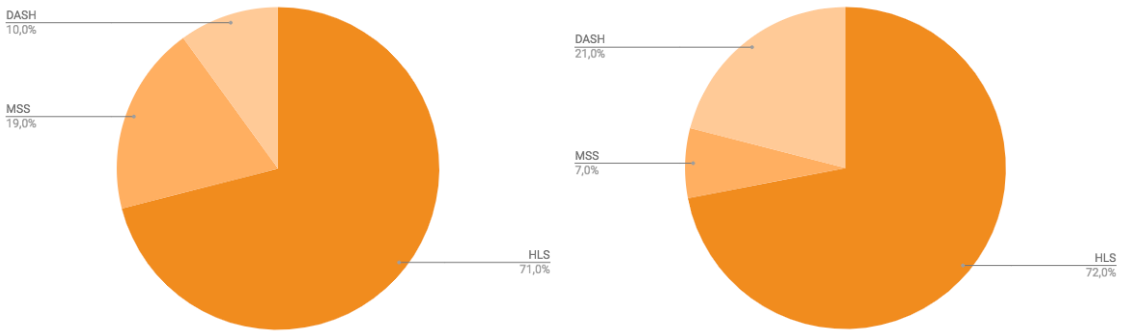
3.3.5.2 Aspetos Técnicos

	MPEG-DASH	**HLS**	**MSS**	**HDS**
Tipo	Aberto, *standards-based*	Proprietário	Proprietário	Proprietário
Codecs de origem (vídeo)	Agnóstico	H.264	H.264, VC-1	H.264, VP6
Codecs de origem (aúdio)	Agnóstico	AAC, MP3	AAC, WMA	AAC, MP3
Formato do Segmento	Fragmentos MP4 + MPEG-2 TS	MPEG-2 TS	Fragmentos MP4	Fragmentos MP4
Segmentação e distribuição	Múltiplos vendedores standard HTTP	Múltiplos vendedores standard HTTP	Servidor Microsoft IIS (+ outros vendedores)	Adobe Interactive Server (ou Adobe tools + módulo Apache)
Reprodução	3GPP-Rel 9 ou clientes MPEG (+ Helix)	Apple iOS, QT X (+ Helix)	Silverlight (+ Helix)	Flash, Air
Duração do Segmento	Flexível	10 segundos	2-4 segundos	2-4 segundos

Tabela 3.2: Comparação dos aspetos técnicos de cada tecnologia

3.3.5.3 Utilização no Mercado Atual

As tecnologias de *streaming* adaptativo têm vindo a crescer nestes últimos anos devido ao crescimento tecnológico e ao aumento do consumo da Internet. Cada vez mais o utilizador final exige uma qualidade de visualização maior assim como uma eficiência a nível de débito. Atualmente as tecnologias de *streaming* adaptativo competem neste ambiente pela sua adoção no mercado e popularidade.



(a) Análise de 2016 [Enca]

(b) Análise de 2017 [Encb]

Figura 3.11: Resultados da popularidade das três tecnologias nas duas análises de Encoding.com

Segundo a informação recolhida nestes últimos relatórios (2015-2016) realizados pela empresa *encoding.com* sobre o estado das tecnologias de *streaming* adaptativo, observa-se que o protocolo da Apple, HLS, continua a dominar o mercado como o padrão base da indústria(uma diferença de mais de 100% em relação ao resto), enquanto que as outras tecnologias vão tentando ganhar algum terreno. É de realçar que nestes dois anos o MPEG-DASH conseguiu ultrapassar o protocolo da Microsoft, MSS, trocando de valores percentuais com este no final do ano de 2016 relativamente ao início (2015). O protocolo da Adobe parece não constar neste relatório, possivelmente devido à baixa popularidade.

Capítulo 4

Trabalhos Relacionados

A procura e o estudo de trabalhos relacionados com o tema e âmbito desta dissertação foram imprescindíveis para determinar se o trabalho realizado já se encontra explorado e serviu para fortalecer o conhecimento sobre outros processos e métodos de abordar o tema. Os trabalhos aqui referidos foram escolhidos por serem os trabalhos mais recente que abordam vários aspetos essenciais para a solução. O foco destes trabalhos é a inserção de publicidade para o MPEG-DASH e ambos oferecem maneiras distintas e inovadoras de alcançar esse objetivo. Com a análise destes trabalhos é possível retirar certos aspetos fundamentais assim como perceber como a tecnologia subjacente opera para assim ser possível conceber e implementar uma plataforma cujo objetivo é a inserção de publicidade em *streams* de vídeo utilizando a tecnologia de *streaming* adaptativo MPEG-DASH.

4.1 Técnica Inovadora de Inserção de Publicidade para MPEG-DASH

O objetivo deste trabalho é elaborar uma solução inovadora para a inserção de publicidade do lado do servidor que seja compatível com a tecnologia de *streaming* adaptativo, MPEG-DASH. Esta solução não requer qualquer tipo de modificação no que toca ao padrão de MPEG-DASH e não necessita de nenhuma alteração do *player* da parte do cliente.

A abordagem deste trabalho à inserção de publicidade do lado do servidor na *stream* de vídeo, tem como base a manipulação dos metadados contidos no formato que envolve o conteúdo multimédia. Este trabalho foca-se numa parte da estrutura do formato de ficheiros MPEG-4 que é responsável pela sincronização temporal dos segmentos que constituem o conteúdo DASH.

Esta abordagem utiliza a infromação *box* de um segmento denominada de "tfdt"para calcular o valor dos dados dos segmentos a processar numa determinada posição. Desta maneira é possível introduzir segmentos com publicidade a meio da *stream* do conteúdo original.

Para ser possível a introdução de novos segmentos que contêm publicidade sem quebrar o standard DASH é necessário notificar o ficheiro MPD que a *stream* possui mais segmentos que o original, devido aos segmentos extra de publicidade. A estratégia utilizada por este trabalho assenta no mapeamento dos segmentos em *slots* extra que são criados para, no caso de existir a

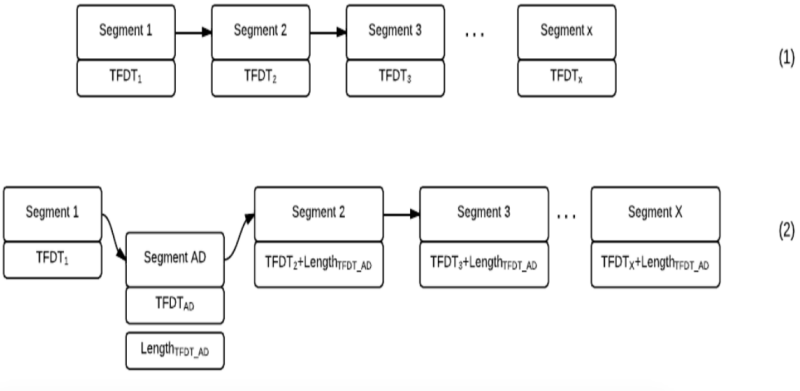


Figura 4.1: Exemplo de sincronização de um segmento. (1) Segmentos que forma um ficheiro de vídeo. (2) Adição dos segmentos de publicidade com o resto dos segmentos originais.[DDH+15]

inserção de publicidade, serem usados para dar lugar a um segmento de conteúdo publicitário. Desta maneira é possível inserir a publicidade dinamicamente em qualquer posição do conteúdo original em tempo real.

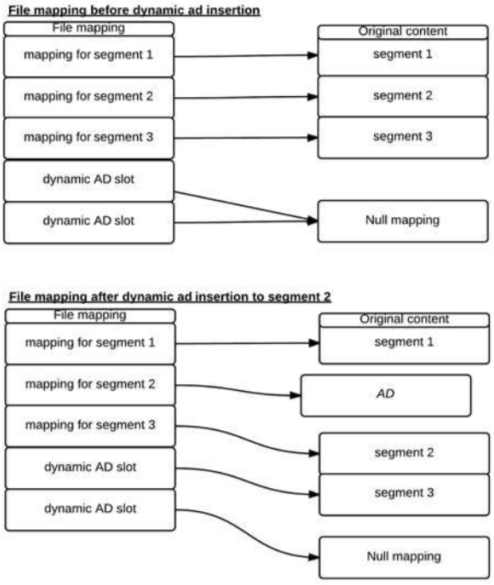


Figura 4.2: Mapeamento dinâmico de segmentos publicitários[DDH+15].

Este trabalho aborda também a questão da exposição do caminho explícito dos segmentos. Ao ser possível distinguir os segmentos de publicidade dos restantes é possível manipular o ficheiro MPD, do lado do cliente, para bloquear ou remover a publicidade. Para resolver este problema, é utilizado um mecanismo de encriptação que oculta o caminho real dos segmentos de maneira a que só o servidor, que conhece a chave de encriptação, seja capaz de realizar a escolha e distribuição dos segmentos corretos [DDH+15].

$$Key = SHA1(Value), Value = username + moviename + random(64bit)$$

Figura 4.3: Função de criação da *hash*[DDH+15].

4.2 Inserção de Publicidade Dinâmica e Personalizada com MPEG-DASH

Este trabalho pretende expor e analisar diferentes métodos de estado da arte para introduzir conteúdo publicitário em *streams* de vídeo utilizando a tecnologia MPEG-DASH assim como realçar certos aspetos no que toda à arquitetura do lado do servidor e também o processo de decisão da publicidade. Neste trabalho também são adicionadas ao *player* de referência, Dash.js, funcionalidades relativas à inserção de publicidade.

Neste trabalho, são analisadas vários padrões de inserção de publicidade tais como o VMAP, VAST, VPAID e VMAG. Estas tecnologias foram desenvolvidas para dar resposta à necessidade de produtores e distribuidores de publicidade de satisfazer os consumidores.

São também expostos e analisados três métodos de inserção de publicidade utilizando os períodos especificados no MPEG-DASH.

- Decisão Imediata de anúncios: Cada pedido do cliente ao servidor para um ficheiro MPD aciona um processo de criação de um manifesto personalizado ao tipo de utilizador. Este método não necessita de recorrer ao mecanismo de XLink pois o ficheiro MPD construído já contém referências diretas para o conteúdo publicitário;

- Sinalização de transição de estado: O gerador do ficheiro MPD coloca XLinks personalizados de acordo com o perfil do utilizador em períodos responsáveis pela inserção de publicidade. Desta maneira o conteúdo que contém publicidade é tratado utilizando o mecanismo de XLink;

- Sinalização de transição sem estado: É mantido um *template* de MPD e a informação do anúncio é codificada no URL do XLink do perídodo correspondente a um anúncio.

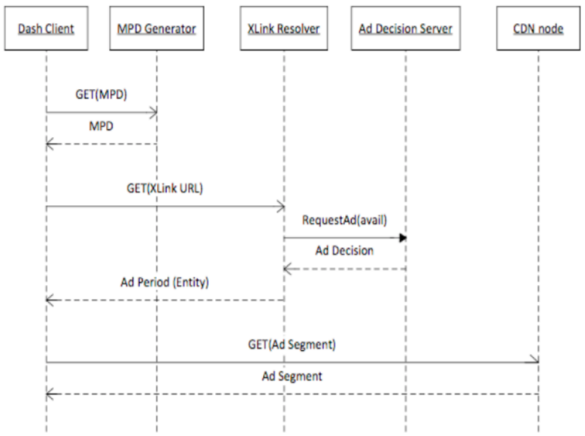


Figura 4.4: Processo de resolução do XLink do lado do servidor[PKSA16].

O processo de decisão da escolha de anúncios é abordado como sendo um aspeto importante mas também tratando-se de um desafio de implementação. Diferentes tipos de dados são referidos para a escolha dos anúncios como a localização do indivíduo, dados gerais sobre o público alvo que podem ser introduzidos pelo produtor da campanha publicitária ou serem recolhidos por uma plataforma automática de análise de dados. Estes dados podem ser enviados juntamente com o pedido de XLink de maneira a obter anúncios diferentes para as mesmas *streams* de conteúdo multimédia.

```
<MPD>
 <Period start="PT0.00S" duration="PT600.6S"
id="movie period #1">
  <AssetIdentifier
schemeIdUri="urn:org:dashif:asset-id:2013"
value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-
8092-1E49-W">
</Period>
 <Period xlink:href="http://se-
mashup.fokus.fraunhofer.de:8080/mws15/
period_timescapes_komp?gender=male"
xlink:actuate="onLoad"></Period>
  <Period duration="PT600.6S" id="movie period
#2">
        <AssetIdentifier
schemeIdUri="urn:org:dashif:asset-id:2013"
value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-
8092-1E49-W">
</Period>
</MPD>
```

Figura 4.5: Ficheiro MPD com um exemplo de um XLink[PKSA16].

Os estudos realizados por este trabalho pretendem avaliar as funcionalidades do *player* Dash.js presentes em diferentes *browsers* para diferentes métodos de inserção de publicidade. Desta maneira é possível obter uma visão geral das capacidades de cada *browser* no que toca à distribuição de publicidade em *streams*.

Browser	dash.js 1.5 sample apps			
	Inline reload	Inband reload	XLink	Custom event SCTE-35
Chrome 45.0.2454.92	✓	✓	✓	✓
Safari 8.0.8	×	×	×	×
Firefox Nightly 43.0a1	✓	✓	✓	✓
Edge 20.10240	✓	✓	✓	✓
Opera 33.0.1990.115	✓	✓	✓	✓

Figura 4.6: Suporte das diferentes tecnologias em *browsers* diferentes[PKSA16].

A partir deste trabalho é possível concluir que a inserção de conteúdo publicitário continua a ser um dos grandes desafios e atualmente existe um grande esforço no sentido de criar processos de interoperabilidade entre *browsers* no que toca à reprodução de conteúdo DASH assim como a inserção de anúncios personalizados.

Capítulo 5

Solução

Esta solução está inserida num projeto criado em parceria pela MOG Technologies e a Mirriad, denominado de Valence. Combinando a plataforma de criação de anúncios nativos da Mirriad com as tecnologias e especialidade da MOG em distribuição de conteúdo multimédia, pretende-se desenvolver uma solução inovadora que consiste numa plataforma *cloud* de inserção de publicidade nativa e contextualizada que utilize técnicas de *streaming* adaptativo. Para o cliente final, isto significa que enquanto visualiza um conteúdo audiovisual este será exposto a publicidade embutida no conteúdo de maneira a que esta se adeque à sua localização e ambiente. Por exemplo, duas pessoas de locais diferentes do mundo poderão estar a visualizar o mesmo conteúdo, mas partes dele que correspondem ao *product placement* serão diferentes devido à sua localização. O utilizador do lado do anunciante deverá conseguir inserir os segmentos publicitários na plataforma assim como o conteúdo de maneira simples e eficiente.

A aplicação deverá receber a informação dos anúncios cedida pela empresa de visão por computador, Mirriad, assim como o conteúdo onde estes deverão ser inseridos. Após a inserção nativa dos anúncios no conteúdo, estes serão armazenados num servidor da MOG onde, recorrendo ao protocolo MPEG-DASH, irá ser feito a segmentação dos mesmos de acordo com a localização temporal dos anúncios. Aquando da reprodução de um vídeo, o cliente deverá utilizar o manifesto de MPEG-DASH para efetuar os pedidos necessários dos *chunks* das diferentes qualidades dependendo da sua localização. O processo de seleção das campanhas deverá ter em atenção a localização, características e outros fatores do utilizador que otimizem o impacto das campanhas publicitárias.

5.1 Arquitetura do Valence

A arquitetura da solução descrita neste capítulo insere-se numa arquitetura geral do projeto Valence que a envolve. Isto é, esta solução, apesar de indispensável, é apenas uma parte da arquitetura total do projeto. Existem outros componentes e processos que contribuem para a solução

final desejada por ambas as empresas.

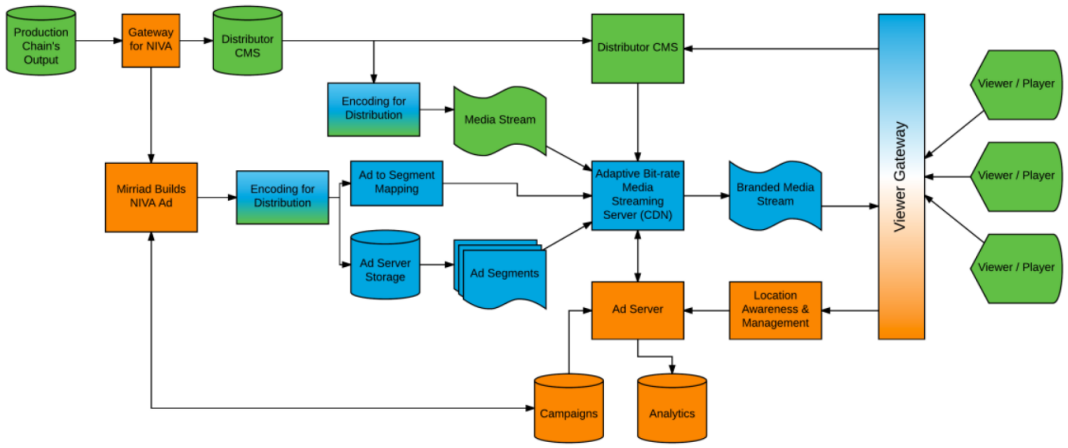


Figura 5.1: Arquitetura geral do projeto Valence.

Na figura 5.1 os componentes a azul, são os componentes responsáveis pela MOG Technologies, os laranjas correspondem aos responsáveis pela Mirriad e a verde pelo produtor de conteúdo. Os componentes que são abordados na solução deste trabalho são os que correspondem aos da MOG Technologies. Esta parte da arquitetura é responsável pela inserção do conteúdo publicitário, armazenamento do conteúdo de *streaming* adaptativo e a sua distribuição até ao cliente final. Como se pode verificar pela figura este é o bloco central de toda a arquitetura, pelo que é indispensável e requer uma implementação cuidada.

5.2 Ferramentas

Ao longo do trabalho, certas ferramentas foram sugeridas para serem utilizadas no sentido de ajudar no desenvolvimento em certas áreas do projeto que não são tão relevantes no que toca aos objetivos do mesmo, como, por exemplo, a codificação de vídeo e áudio e a geração de conteúdo MPEG-DASH. Estes processos fazem parte da solução final, mas são delegados a estas ferramentas. Para recolher informação sobre as ferramentas existentes que se adequam a este projeto e consequentemente formular uma decisão, foi necessário realizar um estudo das ferramentas mais relevantes no mercado. Inicialmente, foi feito um levantamento de certas características que mais valorizavam a solução assim como outras que poderiam ser benéficas no futuro. Algumas ferramentas implementam mais funcionalidades que outras, o que as diferencia, potencialmente, a nível de flexibilidade e escalabilidade. Estas características são importantes, no caso da solução em questão, pois permitem que o produto criado a partir desta seja capaz de se adaptar às diferentes necessidades do mercado que possam ocorrer após a implementação. Seguidamente, foi realizado um estudo das várias ferramentas encontradas de maneira a avalia-las de acordo com as funcionalidades levantadas anteriormente. A seguir são apresentadas as ferramentas e o *software* escolhidos para a implementação da solução.

5.2.1 FFMPEG

O FFMPEG (*Fast Forward MPEG*) é um *software open source* multiplataforma que dispõe de muitas funcionalidades necessárias para todo o tipo de tratamento de conteúdo multimédia. É utilizado nos mais diversos projetos desde pequenas a grandes empresas desta área pois permite acelerar o processo de trabalho devido à sua fácil utilização e também ao grande leque de funções que desempenha. Suporta grande parte dos *codecs* de vídeo e áudio utilizados hoje em dia, assim como outros que são necessários em projetos mais especializados. Por ser *open-source* traz também a vantagem de ser possível expandi-lo de acordo com as necessidades de cada projeto. O código fonte do FFMPEG pode ser adicionado num projeto possibilitando assim o uso das suas funções diretamente a partir do código. Esta abordagem é útil para quem pretende adicionar alguma funcionalidade extra ao código original ou simplesmente para o integrar num pedaço de *software* (biblioteca). Devido à utilização do FFMPEG neste trabalho não ser muito extensa (é apenas necessário para a codificação e aplicação de vídeo sobre um outro), este foi apenas utilizado o *software* como uma ferramenta de linha de comandos. Para além do uso da biblioteca que compõe o FFMPEG este pode também ser utilizado como um comando o que para o caso de uso mais simples basta. A estrutura completa da sintaxe do comando FFMPEG, assim como todas as suas funções não serão aqui detalhadas, porém, no comando seguinte pode ser vista sua estrutura geral que servirá de base para os comandos mais complexos.

```
ffmpeg [global_options] {[input_file_options] -i input_url} ... {[output_file_options] output_url} ...
```

Figura 5.2: Estrutura geral de um comando da ferramenta FFMPEG.

5.2.2 MP4Box

A MP4Box é uma ferramenta incluída no GPAC, um projeto multimédia *open source* não só utilizado em investigação e trabalhos académicos mas também em projetos da indústria. O MP4Box permite, entre outras funções, manipular vários tipos de ficheiros de multimédia ISO (MP4, 3GP), encriptar *streams*, converter vários formatos (AVI, MPG, TS) multimédia em ficheiros ISO e criar conteúdo de *bitrate* adaptativo HTTP. Este *software* não efetua qualquer codificação no conteúdo em si, sendo assim, essencialmente, uma ferramenta de empacotamento de conteúdo multimédia. A MP4Box, similarmente ao FFMPEG, é utilizada recorrendo também à linha de comandos.

```
MP4Box -dash 1000 file.mp4
```

Figura 5.3: Exemplo de um comando utilizando a ferramenta MP4Box para criar conteúdo DASH com 1 segundo de duração dos segmentos.

5.2.3 Dash.js (Video Player)

O Dash.js é um reprodutor de vídeo *open source* criado e mantido pela DASH Industry Forum que tem como objetivo fornecer uma base para *players* de vídeo e áudio que pretendam reproduzir conteúdo compatível com o padrão MPEG-DASH. Este *player* utiliza as extensões da API Media Source, um conjunto de ferramentas para a manipulação de elementos de multimédia em navegadores de internet. Esta API permite controlar o fluxo de *streams* de vídeo e a sua reprodução, utilizando a linguagem javascript diretamente no *browser* o que elimina a necessidade de *plugins* de terceiros para o mesmo efeito. Este *player* foi escolhido para testar a reprodução dos vídeos gerados na solução deste trabalho por se tratar de uma ferramenta *open source*, robusta, que implementa várias funcionalidades indispensáveis para o projeto e também por prometer ser uma implementação fiel do padrão DASH.

5.2.4 Comparação das Ferramentas Estudadas

Para a comparação das ferramentas estudadas foi criada uma tabela de comparação em que cada linha é uma funcionalidade considerada importante para a aplicação desenvolvida. O símbolo *x* indica que a ferramenta possui essa funcionalidade e o símbolo *?* representa a falta de informação ou de testes dessa funcionalidade. Esta tabela é referente apenas à criação de conteúdo DASH por cada ferramenta. A codificação e outros aspetos não foram estudados com o mesmo nível de detalhe.

Funcionalidades Relevantes	**FFMPEG**	**MP4Box (GPAC)**	**avconv (fork FFMPEG)**	**DASHEncoder (x264 + MP4Box)**
Geração Dash	x	x	x	x
Live	x	x	x	x
Criação dos segmentos	x	x	x	x
Criação do MPD	x	x	x	x
Configuração da duração dos segmentos	x	x	x	x
Template no MPD para listas de segmentos	x	x	x	?
Timeline dos segmentos	x	x	x	?
Segmentos num ficheiro apenas	x	x	x	?
Alocar streams a AdaptationsSets (ex: separar vídeo e áudio)	x		x	
Geração e controlo de períodos		x		?
Tempo mínimo do buffer		x		x
Controlo sobre o stream switch		x		?

Tabela 5.1: Comparação das diferentes ferramentas na criação de conteúdo DASH

Depois da análise feita às diferentes tecnologias, foi decidido começar por implementar a solução, no que toca à geração de conteúdo DASH, recorrendo à ferramenta MP4Box, que apresentou uma maior flexibilidade e quantidade de características pertinentes para o projeto.

Processos/Configurações	**1**	**2**	**3**	**4**
Codificação	FFMPEG	x264	DASHEncoder	FFMPEG
Geração de DASH	MP4Box	MP4Box		(WebM, libvorbis + libvpx)
Player	Dash.js			

Tabela 5.2: Diferentes configurações para a implementação

De entre todas as diferentes configurações possíveis para a solução presentes na tabela 5.2, a escolhida foi a primeira, uma codificação em FFMPEG com a construção do conteúdo DASH pela MP4Box e de seguida a visualização do conteúdo no *player* Dash.js. No que toca ao *codec*, o FFMPEG é idêntico à ferramenta x264, aliás, este utiliza uma versão desta para a codificação em H264. Sendo assim, torna-se mais vantajoso utilizar o FFMPEG que é mais extenso para o processo de codificação, mas também porque oferece à aplicação mais flexibilidade.

5.3 Arquitetura da Solução

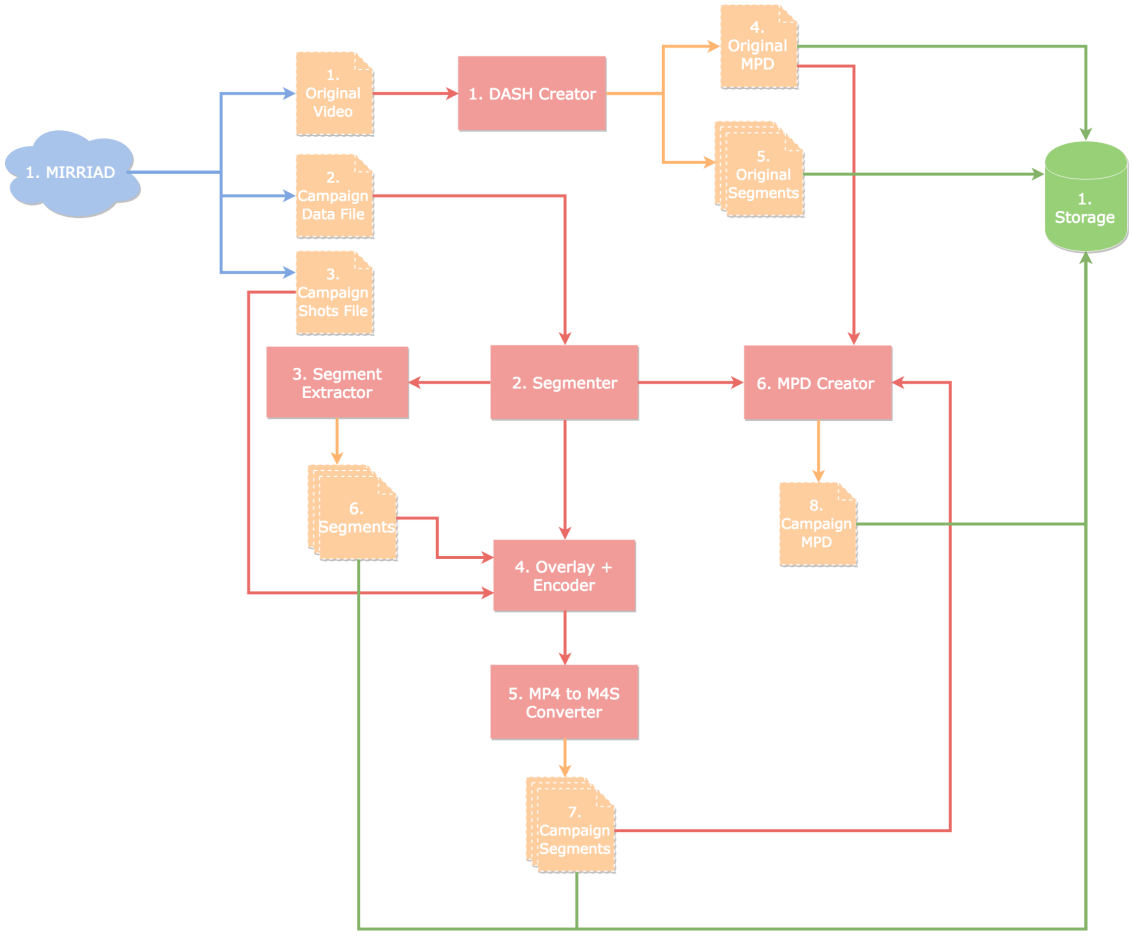


Figura 5.4: Esquema da Arquitetura da Solução.

A figura 5.4 representa a arquitetura genérica da solução desenvolvida neste trabalho. Os elementos a vermelho representam os componentes do sistema; os amarelos, os ficheiros necessários assim como os gerados pelo sistema; a verde o local de armazenamento e a azul o local de origem dos ficheiros iniciais.

5.3.1 Componentes

Aqui se enumeram os componentes, presentes na figura 5.4, que compõem a arquitetura do sistema e que são responsáveis pelo processamento da informação assim como os processos e transformações necessárias para aplicar uma campanha publicitária num determinado vídeo de origem.

- 1. DASH Creator: Responsável pela criação do perfil DASH original, ou seja, todos os segmentos de vídeo e de áudio com uma determinada duração e um ficheiro de manifesto MPD que contém referências para cada segmento. Primeiro Codifica o vídeo original e posteriormente chama a ferramenta MP4Box para segmentar e criar o ficheiro de manifesto MPD.

- 2. Segmenter: Com a informação sobre as diferentes cenas fornecidas pelo ficheiro de dados da campanha, este componente identifica quais segmentos têm anúncios e chama o componente Overlay + Encoder com os argumentos necessários para processar um determinado segmento.

- 3. Segment Extractor: Este componente é responsável por criar segmentos, do vídeo original, correspondentes aos segmentos nos quais os anúncios são exibidos.

- 4. Overlay + Encoder: Sobrepõe o ficheiro de vídeo criado no Componente Segment Extractor e codifica-o.

- 5. MP4 to M4S Converter: Converte um ficheiro de formato MP4 fornecido pelo componente Overlay + Encoder num ficheiro de formato MP4 segmentado (.m4s)

- 6. MPD Creator: Cria um MPD relativo à campanha publicitária com os segmentos de publicidade criados até então.

5.3.2 Ficheiros

Para além dos componentes, na figura 5.4 é possível observar que estes dependem de certos elementos, mas também são capazes de os criar. Estes elementos representam todos os ficheiros envolvidos na arquitetura da solução. Serão aqui enumerados respetivamente.

- 1. Ficheiro do vídeo original: o ficheiro de vídeo original utilizado para aplicar as campanhas publicitárias.

- 2. Campaign Data File: O ficheiro que contém todas as informações sobre a campanha, incluindo as referências aos *frames* necessários de cada cena, fps, ...
- 3. Campaign Shots File: Este ficheiro contém todas as cenas publicitárias em modo de transparência, isto é, apenas são visíveis as partes do vídeo onde efetivamente é possível observar o conteúdo da campanha e as partes restantes encontram-se a preto. Estes *frames* são posteriormente aplicados em cima dos segmentos do vídeo original.
- 4. Original MPD: O ficheiro de manifesto (metadados) do DASH contendo os caminhos de segmento não processados.
- 5. Original Segments: Os segmentos originais a serem usados pelo componente Overlay + Encoder (.mov).
- 6. Segments: Os segmentos correspondentes ao vídeo original (.m4s).
- 7. Campaign Segments: Os segmentos criados, que contêm conteúdo publicitário, para cada campanha (.m4s).
- 8. Campaign MPD: O ficheiro de dados de manifesto MPD correspondente a cada campanha. Contém referências aos segmentos originais mas também aos que contêm publicidade.

5.3.3 Armazenamento

Os ficheiros resultantes da execução desta aplicação deverão ser armazenados numa plataforma capaz de servir pedidos HTTP, como por exemplo um servidor *web* Apache. Todos os processos representados na figura 5.4 são gerados apenas para uma campanha de publicidade que está associada a um vídeo original.

5.3.4 Origem

Por defeito todos os ficheiros necessários para o funcionamento desta solução serão fornecidos pela Mirriad já devidamente tratados pela mesma. Tanto o vídeo original como o vídeo das cenas publicitárias e o ficheiro da informação da campanha não são sujeitos a alterações por esta aplicação. Desta maneira existe um desacoplamento entre as duas entidades o que torna todo o processo modular. No entanto, a solução implementada assume que os formatos dos ficheiros inseridos assim como a sua estrutura não se alteram. É o caso do ficheiro da informação da campanha que segue uma estrutura rígida que deve ser respeitada pela entidade que fornece o conteúdo. Para a prova de conceito os ficheiros de vídeo necessários devem ter ambos as mesmas características (*codecs*, *bitrate*, formato de cor e *frames* por segundo) e respeitar o *codec* que a solução implementa (Prores). Isto deve-se ao processo de segmentação desta solução, nomeadamente o *overlay*, pois este requer as mesmas características nos vídeos a processar. Idealmente, a aplicação deverá ser capaz de conseguir detetar o *codec* e as suas características e passa-las para o componente que utiliza o *overlay*.

5.4 Estrutura de Ficheiros

Para a reprodução correta do conteúdo DASH é necessário que exista uma relação entre o ficheiro de manifesto MPD e os segmentos respetivos. No caso da inserção de segmentos de publicidade é necessário utilizar dois tipos de segmentos: os originais e os que contêm a publicidade. O espaço de armazenamento deverá ser capaz de suportar vários ficheiros de campanhas diferentes para cada vídeo original. Isto é, cada diretório correspondente a um vídeo deverá conter outros diretórios onde estão armazenados os ficheiros de manifesto MPD assim como apenas os segmentos que contêm a publicidade. Desta maneira o MPD de cada campanha publicitária consegue saber onde estão os segmentos originais, a partir do MPD original do diretório acima dele, e também os segmentos de publicidade que estão no mesmo diretório.

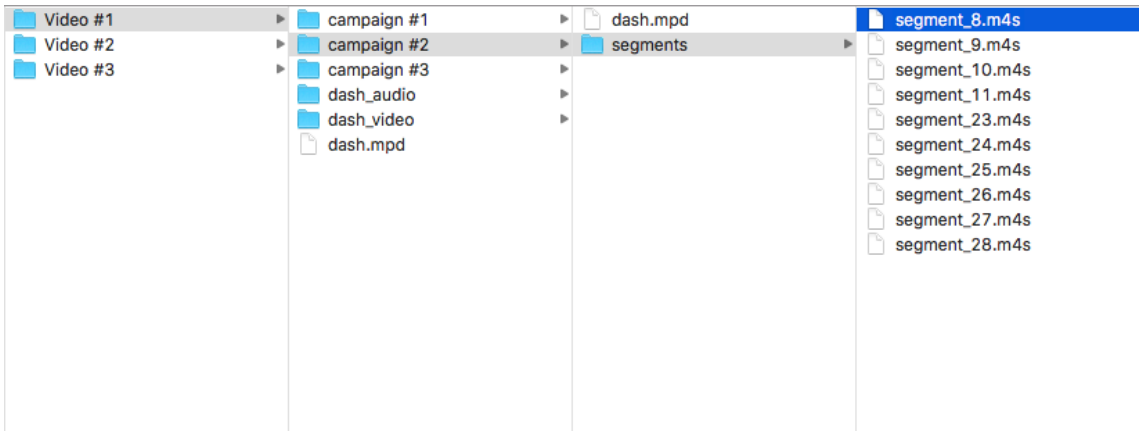


Figura 5.5: Estrutura de ficheiros da solução.

5.5 Estratégia de Segmentação

Para a correta sobreposição das cenas do vídeo da campanha publicitária em cada segmento é necessário ter em conta a duração deste pois pode influenciar a estratégia de segmentação. A duração do segmento é escolhida pelo utilizador, podendo tomar vários valores. Sendo assim é possível existir um desalinhamento entre os cortes correspondentes aos segmentos DASH e as cenas da publicidade. Ou seja, pode existir conteúdo publicitário que se aplica em vários segmentos ou até segmentos com múltiplas cenas de publicidade.

No sentido de implementar uma solução capaz de lidar com este problema foram ponderadas duas estratégias:

- Estratégia Completa: Para cada campanha realizar a sobreposição de todas as cenas publicitárias no vídeo original e e de seguida segmentar o vídeo resultante em segmentos com a duração pretendida. Esta abordagem tem a vantagem de ser simples e de fácil implementação pois não é necessário desenvolver nenhuma lógica extra visto que esta ignora por completo o problema do desalinhamento. No entanto, requer que exista novamente uma sobreposição do vídeo inteiro sempre que se insira uma campanha publicitária com produtos

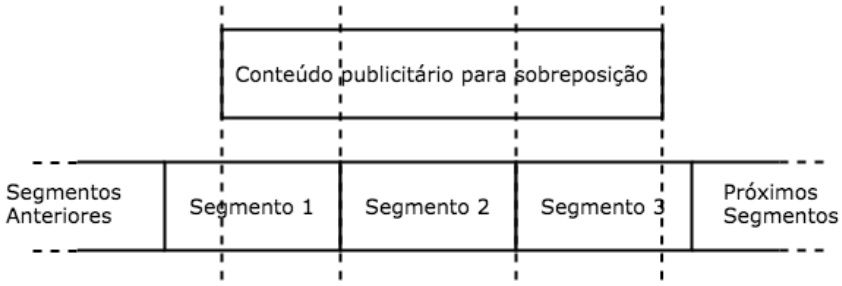


Figura 5.6: Exemplo do desalinhamento entre o conteúdo publicitário e os segmentos.

diferentes. Para vídeos longos ou com uma qualidade elevada, este processo é dispendioso e demorado. Outro problema com esta abordagem é a duplicação dos segmentos que não contêm publicidade em cada campanha, que pode ser tratado recorrendo a um mecanismo idêntico ao da estratégia seguinte e eliminando os segmentos dispensáveis.

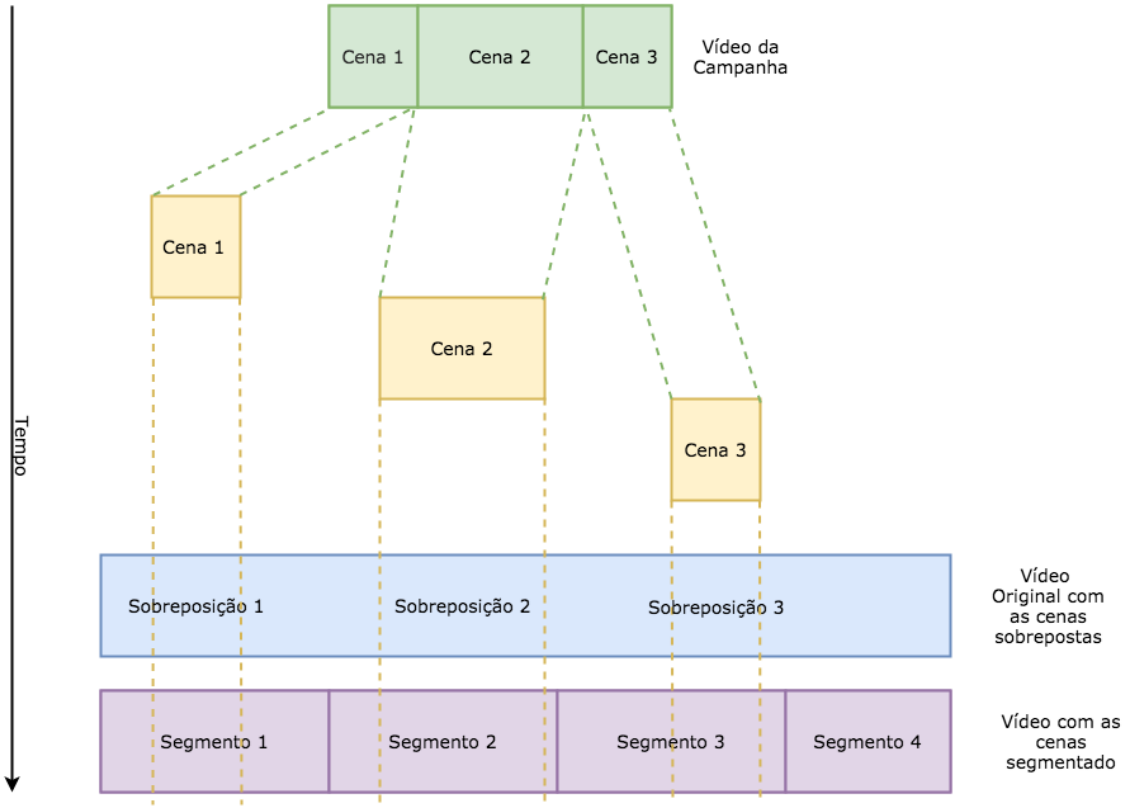


Figura 5.7: Diagrama que representa a estratégia completa.

- Estratégia Parcial: Esta estratégia consiste em processar apenas as partes relevantes do vídeo, as que contêm o material publicitário das campanhas, tratando-se assim de uma abordagem mais sofisticada que a anterior. É mais complexa que a estratégia anterior não só porque necessita de uma lógica capaz de tratar do problema do desalinhamento, mas também porque tem de existir um mecanismo que junte os segmentos publicitários criados aos

segmentos originais do resto do vídeo num único MPD. Esta estratégia pode ser implementada de duas maneiras distintas que estão detalhadas a seguir.

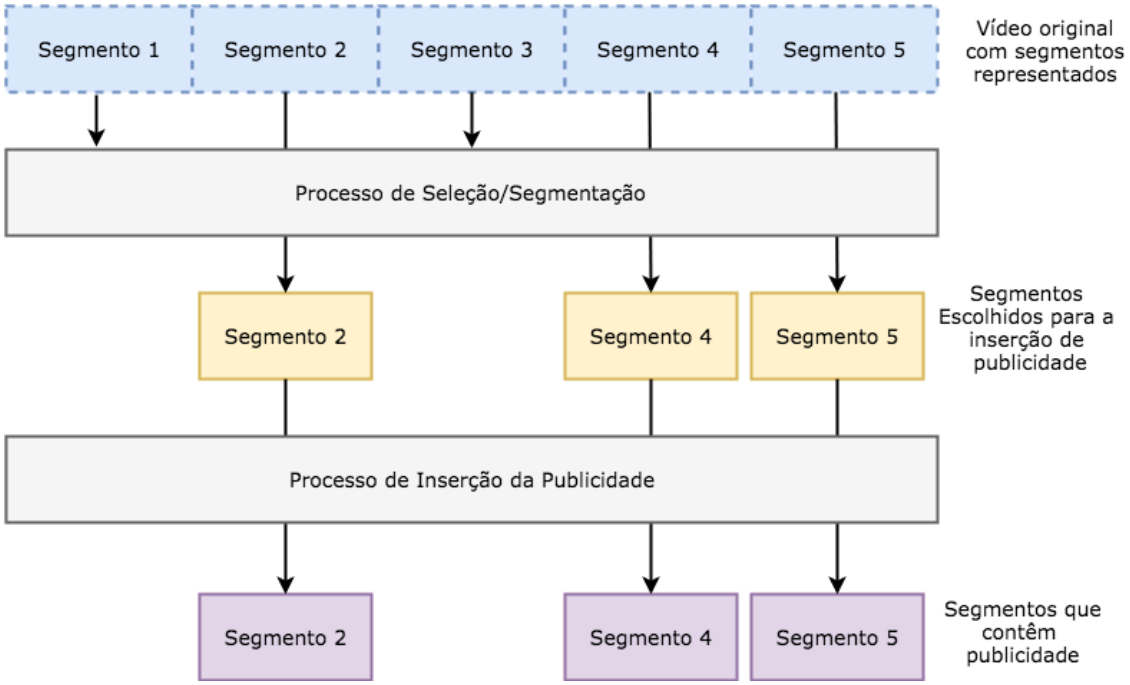


Figura 5.8: Diagrama que representa a estratégia parcial.

– Segmentação e Sobreposição (SS): Nesta abordagem, o primeiro passo é a extração apenas dos segmentos necessários para a campanha (segmentos do vídeo original que contêm, no mínimo, uma cena de publicidade) e, de seguida, a sobreposição do conteúdo publicitário em cada um desses segmentos.

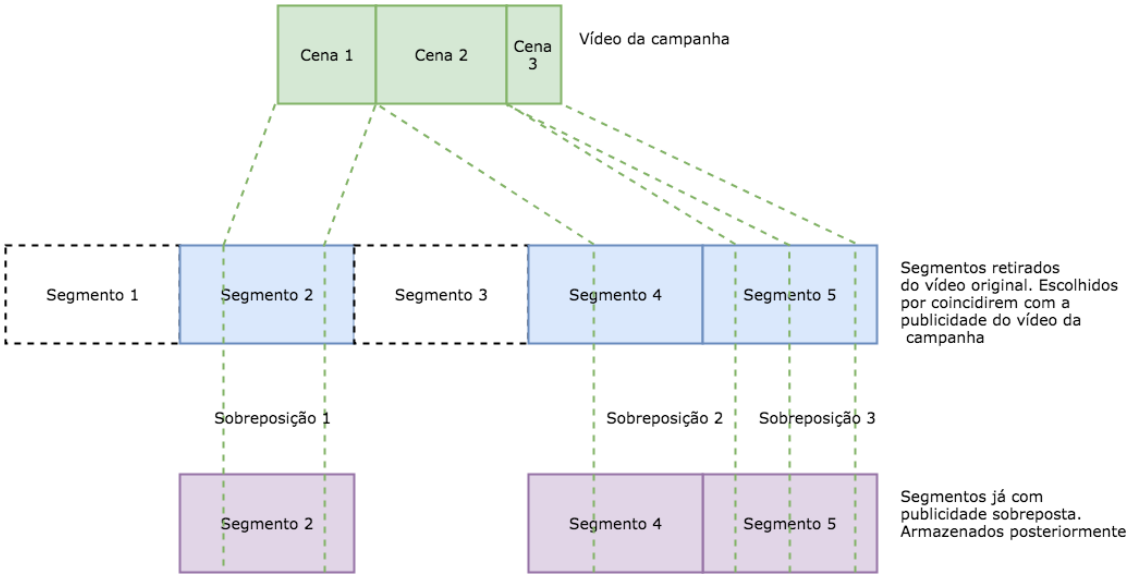


Figura 5.9: Diagrama que representa a estratégia parcial de Segmentação e Sobreposição (SS).

– Extração, Sobreposição e Concatenação (ESC): Este processo, como o nome indica, pode ser descrito em três momentos distintos. Primeiramente, é feito uma extração das partes do vídeo original que contêm apenas os *frames* da publicidade e de seguida é feita a sobreposição da mesma. Por último é construído o segmento recorrendo às partes criadas anteriormente e aos cortes do vídeo original que não contêm publicidade. Entre todas, esta é a abordagem mais complexa devido ao número elevado de operações envolvidas nesta (Segmentação, sobreposição e concatenação de vídeo), e por necessitar de uma camada extra de lógica responsável pela determinação das partes de um segmento que contêm ou não publicidade.

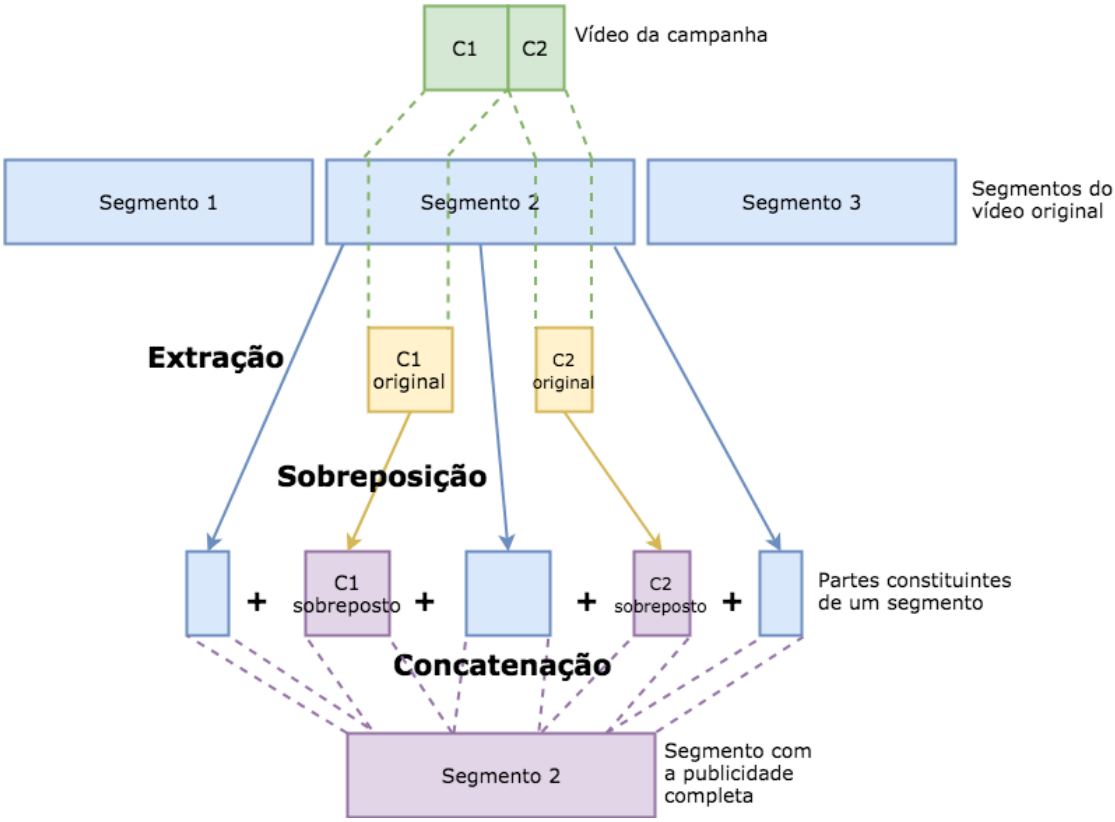


Figura 5.10: Diagrama que representa a estratégia parcial de Extração, Sobreposição e Concatenação (ESC).

5.5.1 Operações em Vídeo

Para a implementação desta solução, no caso das operações em vídeo, foi necessário recorrer a certas funcionalidades da ferramenta FFMPEG que oferecem uma camada de abstração no que toca ao processamento e manipulação de ficheiros de vídeo. Porém, a sintaxe dos comandos necessários para a utilização destas operações não é necessariamente trivial. Os comandos a seguir apresentados pertencem apenas à solução de Segmentação e Sobreposição.

5.5.1.1 Segmentação

Este comando é responsável pela segmentação do vídeo original num único segmento. O segmento originado é depois utilizado para a sobreposição do conteúdo publicitário das campanhas que contêm anúncios inseridos nestes.

Comando de segmentação: "ffmpeg -y -ss <start_second> -i <original_file> -vframes <number_frames> -an -c:v copy <output>":

- <start_second>: O tempo, em segundos, correspondente ao primeiro *frame* de segmentação no vídeo original;
- <original_file>: O caminho do ficheiro de vídeo original;
- <number_frames>: O número máximo de *frames* para a segmentação. Corresponde à duração do segmento DASH em *frames*;
- <output>: O caminho do segmento gerado pela segmentação.

5.5.1.2 Sobreposição e Codificação

O comando seguinte é responsável pela sobreposição assim como a codificação. Isto é, o vídeo é codificado para o codec H.264 ao mesmo tempo que este é sobreposto.

Comando de sobreposição e codificação: "ffmpeg -i <input> -itsoffset <offset> -i <campaign_file> -filter_complex "overlay=enable='between(t,<lower_bound>,<upper_bound>)'-vframes <number_frames> -c:v libx264 -movflags faststart -y -f mp4 <output>":

- <input>: O ficheiro original para ser sobreposto. Corresponde a um segmento;
- <offset>: O tempo em segundos do início do conteúdo da campanha a sobrepor no segmento. Serve para alinhar corretamente os dois vídeos;
- <campaign_file>: O ficheiro com o conteúdo publicitário a sobrepor;
- <lower_bound>: O tempo, em segundos, correspondente ao primeiro *frame* do segmento onde começa o conteúdo publicitário.
- <upper_bound>: O tempo, em segundos, correspondente ao último *frame* do segmento onde começa o conteúdo publicitário.
- <number_frames>: O número de *frames* para a sobreposição. Corresponde ao número de *frames* do segmento DASH. Se o conteúdo publicitário contiver um número menor que este, devido a estar em dois segmentos seguidos, por exemplo, o FFMPEG apenas utiliza os *frames* da campanha correspondentes aos do segmento.
- <output>: O caminho do segmento gerado pela segmentação.

$$baseMediaDecodeTime = ndice \times baseTime$$

Figura 5.11: Fórmula que calcula o valor do *baseMediaDecodeTime*.

5.5.2 Conversão de Formatos

O *player* utilizado para a reprodução do conteúdo final, que será uma mistura entre segmentos relativos ao conteúdo original e segmentos aos quais foram adicionados as camadas de publicidade, não é capaz de reproduzir sequencialmente ficheiros normais de formato MP4 e apenas suporta ficheiros segmentados de formato MP4 fragmentados. O *player* recolhe alguns dados dos ficheiros de segmento, entre eles o *baseMediaDecodeTime*, que é responsável pelo alinhamento da linha temporal. Se um segmento não contém nos seus cabeçalhos o valor correspondente ao *time stamp* até então construído no *player*, este não é capaz de reproduzir o conteúdo corretamente. Para inserir um segmento fora da sequência normal dos segmentos, por exemplo trocar o primeiro segmento pelo último, é necessário alterar este valor de acordo com o índice do segmento.

A ferramenta utilizada neste processo para a criação dos ficheiros necessários para a reprodução DASH, a MP4Box, cria ficheiros com a extensão m4s, o que não se encontra entre as extensões definidas no padrão MPEG-4 Part 14. Após a análise extensa da estrutura dos segmentos criados pela ferramenta, foi possível concluir que estes são ficheiros MP4 fragmentados, em que cada ficheiro segmento é uma parte do ficheiro MP4 geral fragmentado. De facto, é possível criar os ficheiros para o *streamming* de DASH, com esta ferramenta, de maneira a que exista apenas um ficheiro MP4, sendo este um ficheiro MP4 fragmentado. O padrão de DASH suporta este formato, pois é possível, através de limites explícitos de *bytes*, saber onde se encontra cada segmento no ficheiro. Para a solução implementada, esta estrutura não é a melhor, pois estando cada segmento no mesmo ficheiro as operações para a escrita dos segmentos de publicidade seriam necessárias cada vez que se mudasse de campanha publicitária. A abordagem dos vários segmentos separados em múltiplos ficheiros pareceu ser a mais flexível, fácil de implementar e é mais adequada para testar.

Após ser decidida qual a estratégia de ficheiros MP4 utilizar, foi criado uma ferramenta capaz de converter um ficheiro MP4 normal num ficheiro MP4 fragmentado e segmentado(.m4s). Esta conversão é necessária pois os ficheiros de vídeo alterados (segmentos com publicidade) são processados pela ferramenta FFMPEG que cria ficheiros com a estrutura normal de ficheiros MP4. Para além disso é necessário, como já foi referido no parágrafo anterior, escolher um índice do segmento assim como um *timestamp* de maneira a calcular onde este se insere na linha temporal do vídeo completo.

5.5.2.1 Formatos MP4

O MPEG-4 Part 14, ou mais conhecido por MP4, é um padrão que especifica um *container* de conteúdo multimédia, isto é, caracteriza um formato de ficheiro capaz de conter várias *streams* de

vídeo e de áudio, suportar legendas nativas, imagens entre outros. É um formato muito utilizado na *internet* para a partilha e *streaming* de vídeo. O MP4 permite utilizar vários *codecs* em cada *stream* de vídeo como MPEG-2, H.264/AVC e H.265/HEVC. A ferramenta utilizada neste trabalho, a MP4Box, tem como comportamento padrão a segmentação de ficheiros MP4 normais em vários segmentos MP4 fragmentados. Para a implementação da transcodificação necessária para a transformação de um formato para o outro foi necessário estudar a estrutura interna de um ficheiro MP4 normal assim como o de um fragmentado.

A estrutura dos ficheiros MP4 é definida pelo *ISO base media file format* ou MPEG-4 Part 12. Cada ficheiro de MP4 está dividido em várias caixas que contêm informação relativa às streams de conteúdo multimédia. A informação de metadados contida nas caixas é representada por palavras (4 bytes) por motivos de facilidade de leitura dos dados. Cada caixa tem, no seu cabeçalho, um identificador que determina o tipo da mesma (ex: "fytp"para informação do tipo de ficheiro, "mdat"para os conteúdos de multimédia, ...)[W3C].

Figura 5.12: Estrutura de uma *box*.

A estrutura de um ficheiro normal MP4 difere da estrutura de um ficheiro fragmentado em diversos aspetos. De um modo geral o ficheiro fragmentado contém o mesmo conteúdo que um ficheiro normal, mas ao contrário do primeiro, este separa-o em múltiplas partes, em *boxes*, que contêm o conteúdo correspondente e os seus respetivos metadados.

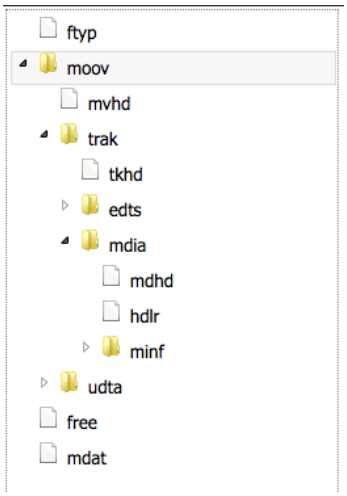


Figura 5.13: Estrutura de um ficheiro MP4 não fragmentado. Algumas *boxes* não se apresentam aqui totalmente expandidas.

- Ftyp Box: Esta *box* é obrigatória e deve ser a primeira a aparecer no ficheiro. Indica qual o tipo de ficheiro e os diferentes formatos em que este é compatível (*brands*);
- Moov Box: Responsável por armazenar todos os metadados do conteúdo (duração, escala temporal, localização dos *frames* e o seu tamanho,...). Esta *box* tem a sua própria estrutura, e está organizada de acordo com a figura 5.13;
- Mdat Box: Contém todo o conteúdo de multimédia, isto é, todos os *streams* (todos os *frames* de vídeo, áudio,...) adjacentes uns aos outros sem qualquer informação adicional de referência(estes dados estão reservados para a *box* "moov").

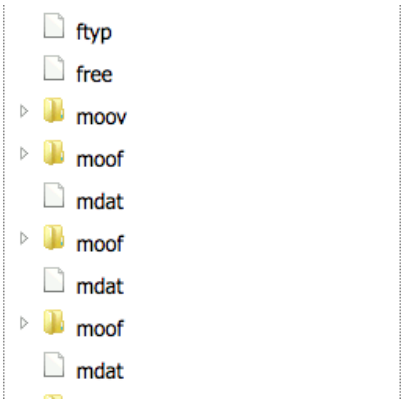
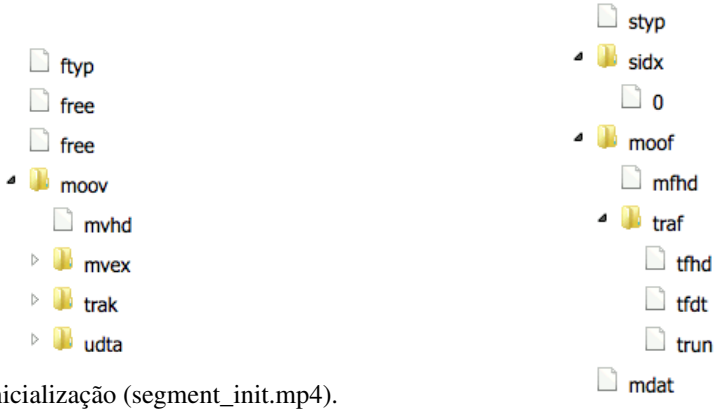


Figura 5.14: Estrutura de um ficheiro MP4 fragmentado.

Como se pode verificar pela figura 5.14, a estrutura do ficheiro MP4 fragmentado é bastante idêntica ao do ficheiro de MP4 normal no que toca às primeiras caixas ("ftyp"e "moov"), no entanto, este difere na organização do conteúdo, como seria de esperar. Isto é, para cada fragmentação do conteúdo principal existe uma *box* "moof", responsável pela indexação, e uma *box* "mdat"que contém o conteúdo multimédia respetivo.



(a) Ficheiro de inicialização (segment_init.mp4).

(b) Segmento MP4 (segment_index.m4s).

Figura 5.15: Estrutura dos ficheiros MP4 usados em conteúdo DASH.

Os segmentos criados pela ferramenta MP4Box podem ser vistos como partes de um ficheiro MP4 fragmentado. Cada segmento contém um par de caixas, uma *box* "moof"e outra *box* "mdat"que corresponde a um dos fragmentos do ficheiro MP4 fragmentado. Ao contrário dos outros formatos MP4, este começa com uma *box* "styp"que sinaliza o ficheiro como sendo um segmento. Para além dos ficheiros de segmento é necessário existir um ficheiro de inicialização (Normalmente designado de "segment_init.mp4") que contém os dados referentes a um ficheiro fragmentado como se fosse um ficheiro completo (informação relativa ao *codec* utilizado, alinhamento das linhas temporais e identificação da faixa de reprodução em segmentos com várias faixas de vídeo, áudio entre outros).

5.5.2.2 H264

O conteúdo multimédia referente ao vídeo de um segmento MP4 está inserido numa *box* "mdat"que apenas contém o conteúdo multimédia codificado. No caso de um ficheiro codificado com o *codec* H264 estas partes do conteúdo, a que se chamam de amostras (no padrão MP4) ou *frames* (na codificação H264) podem conter dentro deles informação relativa ao codec, como por exemplo, informação referente ao perfil do mesmo e até cabeçalhos em cada *key frame* sinalizando os diferentes tipos de *frame*. Durante a análise de ambos os formatos de ficheiros utilizados pelas ferramentas FFMPEG e MP4Box (ficheiro MP4 não fragmentado e ficheiro MP4 segmentado, respetivamente) foi possível concluir que estes divergem no que respeita à estrutura do conteúdo na *box* "mdat".

Existem vários formatos de *bitstreams* para o armazenamento de conteúdo no padrão H264. No caso do ficheiro MP4 normal a *box* "mdat"contém as *bitstream* que não apresentam códigos iniciais de cabeçalho das unidades NAL de cada *frame*, ao contrário do formato Annex B (códigos 0x000001 ou 0x00000001 dependendo do indicado no campo *lengthSizeMinusOne* da *box* "avcC"que é referente ao tamanho dos códigos, em bytes menos um). Esta estrutura, sem códigos iniciais para cada unidade NAL, denomina-se de AVC1 e é um sub-tipo de estrutura dentro do

padrão H264. É possível determinar se um ficheiro MP4 implementa esta estrutura verificando que este contém o nome de formato (*brand*), neste caso "avc1"na *box* "ftyp".

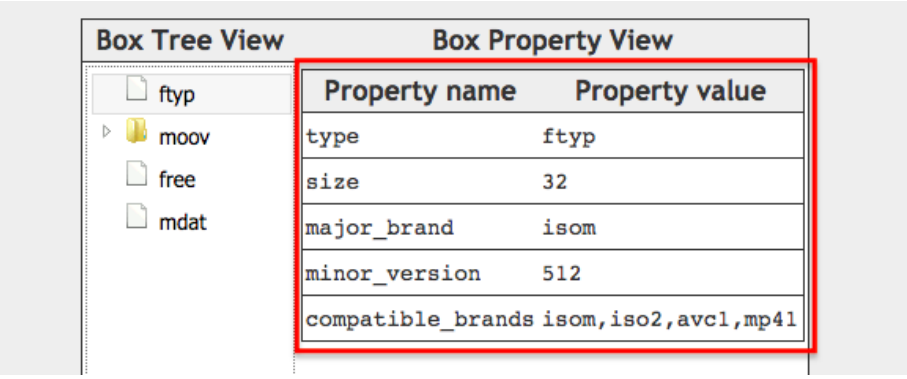


Figura 5.16: *box* do tipo *ftyp* de um ficheiro MP4 que suporta a *brand* avc1.

Na *box* "avc1"pode-se extrair várias informações como, por exemplo, a resolução do vídeo, mas também se consegue retirar o SPS(Sequence Parameter Set) e o PPS (Picture Parameter Set). Cada destas entidades contém informação útil para o descodificador (resolução, frames por segundo, entre outros) e em certas codificações de H264 estes campos aparecem no início de cada *key frame*. Para além de proporcionar ao descodificador uma distribuição em rede mais fiável do conteúdo, também permite alertar-lhe aquando de uma mudança de característica do vídeo na *stream*. O ficheiro MP4 segmentado utiliza esta abordagem pelo que foi necessário ter em conta esta diferença na construção do mesmo.

5.5.2.3 Processo de Conversão

Para a conversão correta entre os dois tipos de formatos utilizados foi necessário conceptualizar e desenvolver uma ferramenta capaz de recolher a informação de um formato, processá-la e armazena-la no formato de saída pretendido. Foi criado um *script* que cria, a partir de um ficheiro MP4 normal, um ficheiro de segmento que pode ser inserido, ou substituído numa *stream* de MPEG-DASH. Para fazer esta conversão o *script* necessita de alguma informação passada como argumentos:

- Caminho do ficheiro MP4 não fragmentado: Este caminho representa o local onde se encontra o ficheiro MP4 que se pretende converter. É importante que ambos os ficheiros de segmento e este ficheiro tenham as mesmas propriedades (número de *frames* por segundo, escala de tempo e um número de *frames* igual ao número de *frames* dos segmentos originais). Para isso é necessário que o perfil de codificação do ficheiro original antes da criação dos segmentos DASH seja igual ao da codificação do ficheiro MP4 não fragmentado.

- Nome base do segmento: Este parâmetro permite a liberdade de escolher o nome base, isto é, o nome de um segmento utilizado para depois acrescentar o seu índice, de maneira a cada ficheiro de segmento ser distinguido pelo sistema de ficheiros.

- Intervalo do *key frame*: É utilizado para detetar qual o *frame* que é do tipo IDR ou I. Desta maneira é possível utilizar a informação do cabeçalho deste para construir uma amostra compatível com o formato MP4 segmentado.

- Número do segmento: O número do segmento, ou o seu índice, é necessário não só para inserir como informação adicional no ficheiro MP4 do segmento (o ficheiro MP4 completo não possui informação do índice pois é tratado como um ficheiro completo), mas também como forma de calcular o valor do *baseMediaDecodeTime* cuja fórmula foi apresentada anteriormente.

- Número de amostras: Este valor é utilizado para percorrer as várias amostras no ficheiro MP4 não fragmentado e também para calcular o tamanho da *box* "trun"que contém o tamanho das amostras.

- Divisor de *key frames*: Indica qual o número de *key frames* presentes no ficheiro de entrada. Em conjunto com o número das amostras e o número do segmento permite calcular a unidade temporal base e a escala de tempo inicial de um segmento.

$$Unidadetemporalbase = \frac{Namostras}{Divisordekeyframes} \times 1000$$

Figura 5.17: Fórmula que calcula o valor da unidade temporal base.

$$EarliestPresentationTime = Unidadetemporalbase \times (Nsegmento - 1)$$

Figura 5.18: Fórmula que calcula o valor do tempo inicial de um segmento.

Por exemplo, se o ficheiro a converter é o 15º segmento, com um intervalo de *key frame* igual a 24 *frames*, a unidade temporal base terá o valor de 24000 e a escala de tempo inicial do segmento terá o valor de 336000. Estes cálculos são necessários para os segmentos gerados corresponderem corretamente com a sequência temporal do resto dos segmentos normais, caso contrário o *player* não é capaz de os reproduzir e encrava.

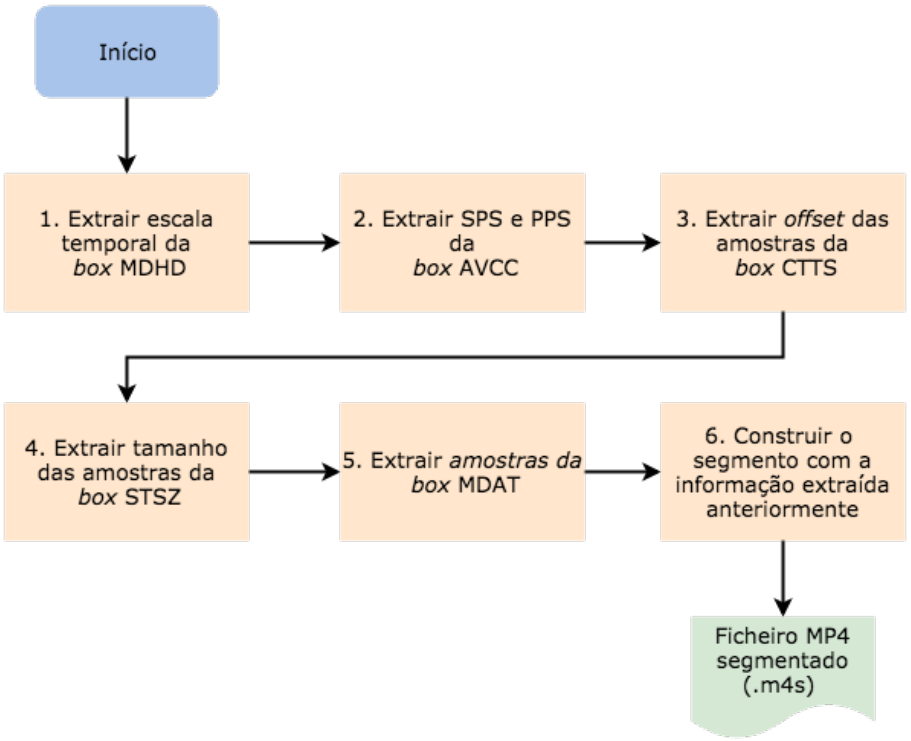


Figura 5.19: Fluxograma que representa o processo de conversão de um ficheiro MP4 normal para um segmentado (.m4s).

A estrutura dos dois tipos de ficheiros são bastante diferentes e para a sua conversão é necessário aplicar as transformações necessárias. A informação dos *frames* encontra-se em sítios diferentes para cada um dos formatos. Sendo assim é necessário extrair essa informação das *boxes* "ctts"e "stsz"e colocá-la na *box* "trun"do formato segmentado. Esta *box* é utilizada pelo *player* para recolher corretamente os *frames* de vídeo para depois os reproduzir. Para além deste processo, é necessário anexar ao início de cada *key frame* a informação recolhida no SPS e PPS.

5.5.3 Ocultação dos Segmentos Publicitários

O MPD gerado a partir dos dois tipos de segmentos (segmentos originais e segmentos relativos a uma campanha publicitária) contém explicitamente o caminho relativo a cada um dos segmentos. Apesar de ser uma abordagem eficaz para a reprodução de conteúdo DASH que contém segmentos em diretórios diferentes, permite, no lado do cliente, a deteção e discriminação dos segmentos de publicidade. Sendo assim, é possível, aquando da receção do ficheiro MPD, substituir os segmentos de publicidade pelos segmentos originais. Basta detetar o caminho relativo dos segmentos originais e aplicar o mesmo padrão, apenas trocando o número do segmento, para os segmentos que contêm publicidade. Desta maneira, e com um *software* especializado, seria possível neutralizar a publicidade dos vídeos consumidos o que é prejudicial para quem beneficia desta.

```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.7.2-DEV-rev438-gc107b22d-master  at 201
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500S" type="static" m
  <ProgramInformation moreInformationURL="http://gpac.io">
    <Title>dash_video.mpd generated by GPAC</Title>
  </ProgramInformation>
  <Period duration="PT0H3M50.481S">
    <AdaptationSet segmentAlignment="true" maxWidth="1920" maxHeight="1080" maxFram
      <Representation id="1" mimeType="video/mp4" codecs="avc3.7A0028" width="1920"
        <SegmentList timescale="11988" duration="47952">
          <Initialization sourceURL=".././dash_video/segment_init.mp4"/>
          <SegmentURL media=".././dash_video/segment_1.m4s"/>
          <SegmentURL media=".././dash_video/segment_2.m4s"/>
          <SegmentURL media=".././dash_video/segment_3.m4s"/>
          <SegmentURL media=".././dash_video/segment_4.m4s"/>
          <SegmentURL media=".././dash_video/segment_5.m4s"/>
          <SegmentURL media=".././dash_video/segment_6.m4s"/>
          <SegmentURL media=".././dash_video/segment_7.m4s"/>
          <SegmentURL media="segments/segment_8.m4s"/>
          <SegmentURL media="segments/segment_9.m4s"/>
          <SegmentURL media="segments/segment_10.m4s"/>
          <SegmentURL media="segments/segment_11.m4s"/>
          <SegmentURL media=".././dash_video/segment_12.m4s"/>
          <SegmentURL media=".././dash_video/segment_13.m4s"/>
          <SegmentURL media=".././dash_video/segment_14.m4s"/>
```

Figura 5.20: Parte do MPD de uma campanha onde se pode visualizar as diferenças nos caminhos dos dois tipos de segmento.

Existem várias abordagens para combater este problema, utilizadas noutras soluções que necessitam de algum mecanismo de ocultação. Uma delas é a encriptação a partir de uma chave, que só o servidor tem acesso, de maneira a ser impossível distinguir quais são os segmentos publicitários, pois todos os segmentos no MPD têm uma *hash* diferente que não permite essa distinção [DDH+15]. Outra é a utilização de um *url* para um *script* no servidor que é responsável pela seleção dos ficheiros de segmento associados a um índice assim como a uma chave da campanha. A abordagem utilizada para esta solução é o uso de *symlinks* (*links* simbólicos) do sistema operativo Linux juntamente com um servidor Apache 2.0. Um *symlink* é um apontador para um ficheiro que pode ter um certo identificador. Neste caso o identificador para os segmentos é um ficheiro com o número do segmento no nome para todos os segmentos e pode apontar tanto para um segmento original como de publicidade. Desta maneira não é possível selecionar um segmento de publicidade pois todos os identificadores partilham do mesmo padrão de nome e são a única maneira de recolher os ficheiros de segmento.

5.6 Geolocalização e Escolha de Campanhas

Uma característica importante na escolha das campanhas é a localização atual do cliente. A partir desta informação é possível alertar o *player* para utilizar uma campanha adequada à geolocalização do utilizador que está a visualizar a *stream* de vídeo. Esta funcionalidade permite às marcas criarem estratégias de *marketing* focadas num público alvo diferente de país para país. Certos produtos poderão ter certas diferenças que façam mais sentido num país do que no outro, ou até podem ser produtos completamente diferentes. Podem existir outros fatores que determinam a escolha de campanhas, como por exemplo a idade, sexo, nacionalidade,... Deste modo, é

imprescindível que a solução implementada suporte este tipo de escolha dinâmica de campanhas. A nível do servidor, onde ficam armazenados os conteúdos de cada campanha, a estrutura de ficheiros já favorece este mecanismo, pois para cada campanha existe um ficheiro MPD que apenas contém a publicidade referente à campanha. Assim, basta o cliente *browser*, ou até o servidor (no caso de plataformas com informação relevante de utilizadores registados), utilizarem a informação recolhida sobre o utilizador para escolherem qual a melhor campanha que se adequa àquele perfil, fazendo então um pedido para o ficheiro MPD correspondente.

5.7 Solução em *Cloud*

No âmbito do trabalho proposto para esta dissertação, era previsto o desenvolvimento de uma solução capaz de funcionar em ambiente *cloud*. A descrição de todos os aspetos da solução anteriormente referida servem como base para o passo seguinte, que é disponibilizar esta como um serviço na *cloud*. Utilizando várias ferramentas próprias para esta aplicação, foi possível transformar a solução linear e baseada num servidor *web* numa outra completamente escalável e paralela. Esta implementação em *cloud* permite a utilização desta aplicação para casos de uso com mais tráfego (escalável) e tem a vantagem de ser possível acelerar o processo de criação da campanha dependendo dos recursos disponíveis.

Esta parte do trabalho realizado foi partilhada com um outro colaborador da MOG Technologies cuja função foi implementar a versão *cloud* definida nesta solução.

5.7.1 Processamento Paralelo

Para ser possível tirar partido do processamento paralelo, foi necessário detetar quais os processos que se comportam de maneira independente, isto é, quais são os processos cujos resultados não afetam incorretamente todo o procedimento geral ou até outros processos. No caso da implementação base, o processamento dos segmentos é feito sequencialmente, mas nenhum segmento depende do anterior ou vice-versa. Isto porque o conteúdo publicitário é tratado de segmento em segmento e não existe qualquer tipo de sobreposição condicional entre eles.

5.7.2 Ferramentas

Para esta solução foram utilizadas ferramentas capazes de criar e organizar aplicações em ambiente *cloud*. A arquitetura presente na imagem 5.4 contém componentes que serão transformados em *containers* completamente independentes e isolados. Esta transformação é necessário pois estes processos poderão estar a correr em máquinas diferentes em diversas redes e necessitam de estar desacoplados do resto da lógica. Este comportamento independente e separado da lógica restante permite um processamento paralelo, em máquinas diferentes, que diminuí em parte o tempo de execução.

5.7.2.1 Docker

O Docker é uma ferramenta de criação e automação de *containers* que permite criar aplicações completamente independentes e isoladas do sistema operativo de onde esta está a ser executada. Ao contrário das máquinas virtuais, as aplicações em Docker, apenas utilizam uma API de comunicação com o *kernel* da máquina hospedeira, o que permite criar um *container* compacto com apenas o necessário para a tarefa relevante. Esta ferramenta é bastante popular e é adotada por muitas empresas do mercado atual pois proporciona um desenvolvimento mais seguro, rápido e sobretudo mais flexível para qualquer aplicação.

```
FROM php:7.2.6-alpine

WORKDIR /segmenter/

RUN apk add --update ffmpeg python py-pip && \
    rm -rf /var/lib/apt/lists/* && \
    rm /var/cache/apk/*

RUN pip install kubernetes

COPY Segmenter.php .
COPY extract_H264.php .
COPY extract.py .

ENTRYPOINT [ "php", "Segmenter.php" ]
```

Figura 5.21: O *dockerfile* utilizado na solução *cloud* que contém o componente Segmenter.

Como se pode observar na figura 5.21, o processo de criação de um *container* é bastante simples e o número de passos para a sua configuração são reduzidos. No caso deste *container*, é utilizado uma imagem do sistema operativo baseado em Linux, Alpine que contém já instalado o PHP que é necessário para a execução dos *scripts*. É ainda feita uma espécie de ponte entre os *scripts* em PHP dos componentes, recorrendo a *scripts* de Python. Estes *scripts* facilitam comunicação entre os *containers*, caso contrário, era necessário configurar manualmente os pedidos HTTP entre eles.

5.7.2.2 Kubernetes

O Kubernetes (K8s) é uma ferramenta *open-source* criada para a orquestração de *containers*. Ou seja, é responsável por todo o processo de automação, gestão de *clusters* de *containers* e a sua manutenção. O Kubernetes permite escalar facilmente aplicações e serviços devido à gestão automática de *containers* e utiliza uma camada API de abstração para gerir os vários *containers* que poderão estar contidos em máquinas diferentes. Devido a estas funcionalidades, esta ferramenta é bastante utilizada juntamente com outras ferramentas de criação de *containers*, como por exemplo o Docker.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: segmenter
  # Note that the Pod does not need to be in the same namespace as the loader.
  labels:
    project: valence
spec:
  restartPolicy: Never
  containers:
  - name: segmenter
    image: user/segmenter
    volumeMounts:
            # name must match the volume name below
            - name: nfs
              mountPath: "/shared"
    args: ["/shared/umg_the_cataracs_missed-u-2.mov", "/shared/dashcreator", "/shared/data.json"]
  volumes:
  - name: nfs
    persistentVolumeClaim:
      claimName: nfs
```

Figura 5.22: Exemplo de um ficheiro de configuração(.yaml) do kubernetes utilizado nesta solução.

5.7.3 Arquitetura e Procedimento

Para tornar a solução em ambiente *cloud* possível foi necessário alterar a arquitetura da solução inicial. Nesta versão, os componentes responsáveis pela transformação dos segmentos originais e inserção de publicidade constituem um único componente em forma de *container*. Isto deve-se ao facto destes componentes poderem ser executados em paralelo utilizando os recursos disponíveis (diferentes máquinas no *cluster*). O resto dos componentes também está incluído num *container* separado para cada um.

No início são lançados os *containers* onde estão inseridos os componentes Segmenter e DASH Creator. Depois de calcular a organização e escolha dos segmentos, o Segmenter é responsável por lançar os *containers* que correspondem a cada segmento que irá conter a publicidade. Enquanto estes *containers* correm lado a lado, o DASH Creator é executado de maneira a criar o conteúdo DASH original. Quando o DASH Creator e os restantes *containers* de segmento acabam a sua execução é então lançado o *container* MPD Creator que trata de criar o MPD para a campanha correspondente.

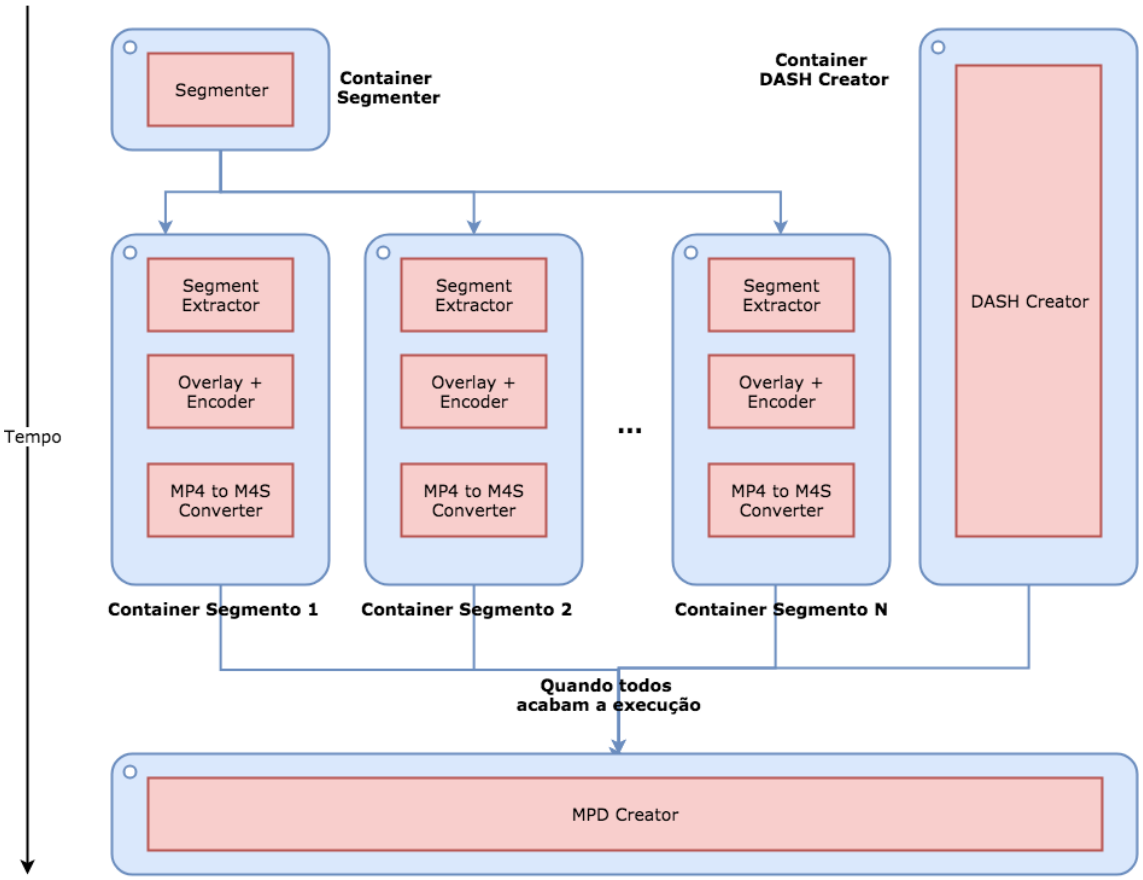


Figura 5.23: Diagrama representante da arquitetura e procedimento da solução *cloud*

Capítulo 6

Resultados e Discussão

A estratégia parcial foi a escolhida para a prova de conceito, pois trata-se de uma abordagem capaz de tratar os segmentos separadamente, o que traz benefícios a nível de paralelização. Este aspeto, no âmbito do trabalho, é bastante relevante, pois trata-se da criação de uma solução capaz de operar num contexto *cloud*. Ou seja, esta estratégia pode tirar partido dos recursos de múltiplas máquinas. Dentro desta estratégia o método de criação dos segmentos escolhido foi o de segmentação e sobreposição por se tratar de uma abordagem mais simples de implementar.

Nesta implementação, foi usado um servidor HTTP Apache 2.0 para o tratamento dos pedidos aos segmentos DASH e o seu armazenamento. Para a visualização do conteúdo DASH foi criada uma *interface* de simples navegação para testar se efetivamente a publicidade foi bem aplicada em todas as partes do vídeo. O conteúdo é reproduzido através do *player* web escolhido para este trabalho, o Dash.js.

Figura 6.1: Interface de visualização das campanhas publicitárias.

Como se pode observar na figura [numero da figura], é possível trocar de campanha, através da lista do lado direito, e visualizar o vídeo respetivo à esquerda. Esta página *web* recolhe a

informação sobre a localização dos ficheiros MPD das campanhas armazenados no servidor *web* assim como a informação relevante da campanha (nome e outras configurações).

Para esta prova de conceito não foi utilizada nenhuma interface gráfica para a inserção de conteúdo publicitário pela parte do produtor de conteúdo pois esta apenas se trata de verificar se todo o processo, desde o tratamento dos ficheiros originais até à visualização de conteúdo DASH pelo cliente, funciona devidamente. Para isso foram utilizados comandos que chamam os *scripts* desenvolvidos para esta solução em vez de uma interface.

(a) Vídeo original. (b) Vídeo com publicidade

Figura 6.2: Diferença entre o vídeo original e o vídeo que contém publicidade.

As imagens da figura 6.2 são retiradas do vídeo original e do conteúdo gerado pela implementação da solução desta prova de conceito. Para validar esta implementação, o vídeo da campanha foi percorrido do início até ao fim várias vezes, com valores de duração de segmentos diferentes, com o intuito de detetar algumas falhas a nível de alinhamento do conteúdo publicitário. Não foram encontrados quaisquer problemas de alinhamento durante a reprodução de vídeo nem nenhum erro a nível do *player*.

6.1 Testes Realizados e Resultados

No sentido de obter algumas métricas ao nível de tempo de execução e utilização de recursos foram efetuados testes à solução sequencial assim como à solução em contexto *cloud*. Estes testes fornecem dados de desempenho da solução de maneira a poder ser comparada com futuras implementações e assim incentivar à sua otimização e melhoramento. O vídeo utilizado assim como a campanha publicitária foram os mesmos para ambos os testes.

Característica	**Dados**
Tamanho em disco	4.79 GB
Duração	3m50.48s
Resolução	1920x1080
Frames por segundo	23.98
Formato	QuickTime (.mov)
Codec	prores
Bitrate	166306 kb/s

Tabela 6.1: Características do vídeo utilizado em ambos os testes.

6.1.1 Estratégia Sequencial

Para esta estratégia foram utilizadas métricas tais como o tempo de execução e a carga de processamento da CPU. Este teste tem em conta dois momentos distintos: a criação do conteúdo DASH original e a segmentação e sobreposição da publicidade em todos os segmentos. Desta maneira é possível separar os dois processos e analisá-los no sentido de procurar melhorar o desempenho geral da solução. Estes testes foram repetidos para durações de segmentos diferentes no sentido de determinar se o desempenho da solução implementada variava para diferentes valores de duração.

Característica	**Informação**
CPU	Intel(R) Core(TM) i7-4790 @ 3.60GHz
RAM	8GiB
Sistema Operativo	Ubuntu (linux)

Tabela 6.2: Máquina utilizada para realizar os testes sequencias.

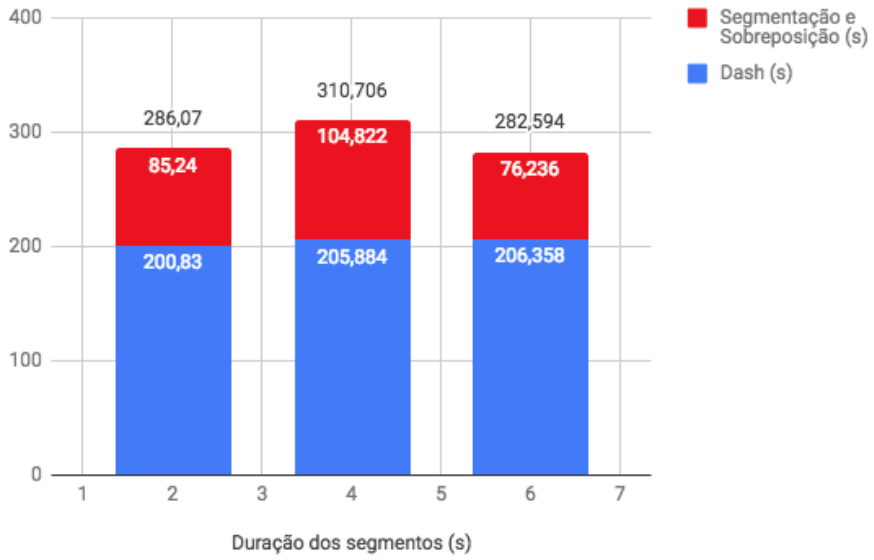


Figura 6.3: Resultados do teste ao tempo de execução dos processos com duração de segmentos diferentes.

Observando o gráfico da figura 6.3 é possível concluir que o tempo de execução da criação de conteúdo DASH não difere muito apesar de aumentar um pouco com o aumento da duração dos segmentos. Já no caso da segmentação e sobreposição é possível observar que existe um aumento de tempo dos dois segundos (85,24s) para os quatro segundos (104,822s) seguidamente de uma descida nos seis segundos (76,236s) que consegue ser mais curto que o primeiro tempo. Este comportamento poderá estar associado ao número de segmentos que diminuí quando a duração dos segmentos aumenta. No entanto é necessário recorrer a mais valores de duração dos segmentos para perceber exatamente qual o comportamento correto e a sua causa.

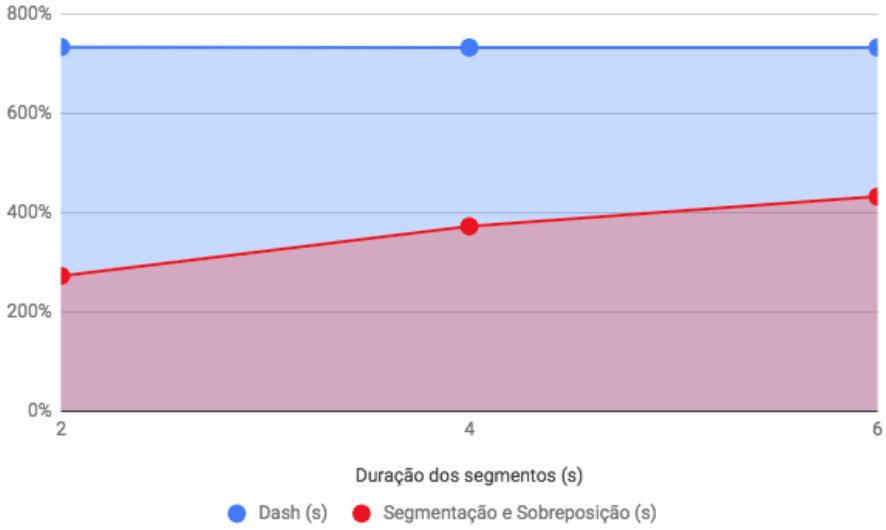


Figura 6.4: Resultados do teste ao uso do CPU dos processos com duração de segmentos diferentes.

Recorrendo à figura 6.4, pode-se concluir que o aumento da duração dos segmentos aumenta a utilização do CPU, neste caso de 50% entre os 2 e os 6 segundos para o processo de segmentação e sobreposição. Este valor pode estar associado ao processo de sobreposição que tende a recorrer mais da CPU com o aumento da duração dos vídeos a serem sobrepostos.

6.1.2 Estratégia Paralela em Ambiente Cloud

O teste feito para a solução em contexto *cloud* apenas utilizou o tempo de execução como dado, visto este ser o mais importante neste caso. Foram utilizadas duas máquinas diferentes para este teste, que fazem parte do mesmo *cluster* de processamento. Para este teste a duração do segmento DASH utilizado foi de 4 segundos.

Switch	**Nome**	**CPU**	**RAM**	**Rede**	**Sistema Operativo**
1 Gbit	Cristal	Intel Core I7-2700K @ 3.4GHz	4 GB	Intel 82579V Gbit	CentOS
	Sagres	Intel Xeon E5640 @ 2.67GHz	32 GB	Intel X550 10 Gbit	CentOS
	Imperial	Intel Core I7-2700K @ 3.4GHz	4 GB	Intel 82579V Gbit	CentOS

Tabela 6.3: Máquinas utilizadas para efetuar os testes da solução em ambiente *cloud*.

	Tempo de Execução
Teste-1	00:03:58
Teste-2	00:03:58
Teste-2	00:04:00
Teste-4	00:03:59
Teste-5	00:03:59
Média	00:03:59

Tabela 6.4: Tempo de execução para os cinco testes realizados.

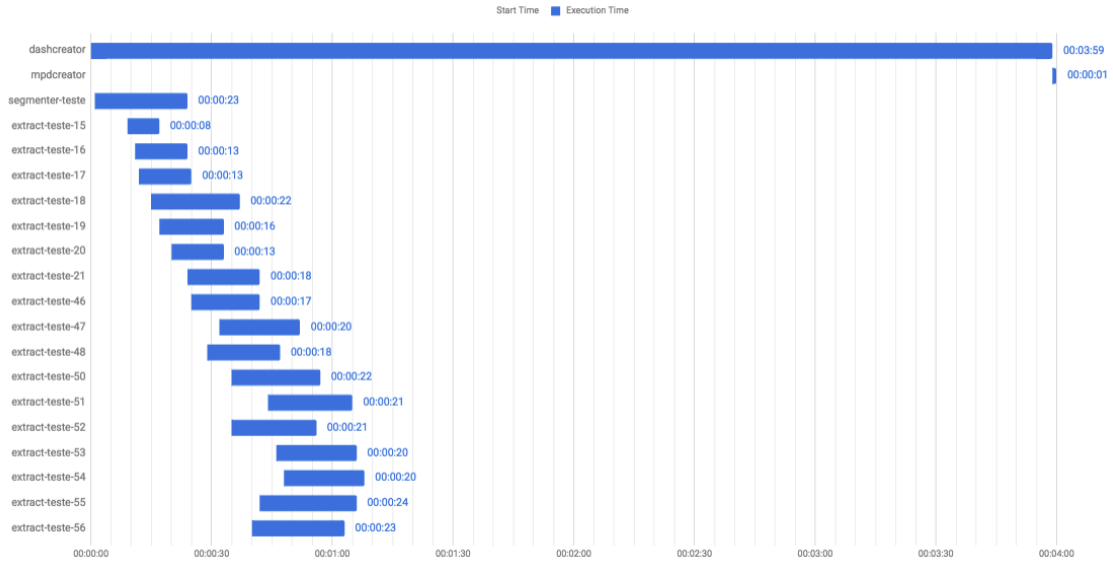


Figura 6.5: Teste realizado do tempo de execução da solução *cloud*.

Pela figura 6.5 é possível constatar que o processo que mais tempo demora a finalizar é o da criação do conteúdo DASH original, o que seria de esperar por se tratar de um ficheiro com uma resolução elevada e muito pesado. A codificação do vídeo original para o codec H.264 é o responsável pela longa duração do processo. Para vídeos com uma duração maior, como por exemplo longas-metragens, este processo poderá demorar horas, dependendo da qualidade do vídeo e outras qualidades que afetam o codificador. Por este motivo, este processo é o gargalo do sistema, no entanto, só se torna um problema na primeira execução do processo, pois para as campanhas seguintes não será necessário criar o conteúdo DASH original, visto que este já se encontra armazenado. Ignorando este aspeto, o tempo de execução da solução *cloud*, neste ambiente de teste, é de apenas 1 minuto e 10 segundos. Este valor poderá diminuir simplesmente com o aumento de máquinas no *cluster* de processamento da *cloud*. Para esta implementação da solução, o número de segmentos é crucial, pois se houver o mesmo número de segmentos que máquinas, a duração do processo será igual à maior duração do processamento de um segmento, que, neste caso, é de 24 segundos, correspondente ao segmento 55.

Capítulo 7

Conclusões e Trabalhos Futuros

A partir do levantamento das tecnologias e soluções existentes assim como o estado da arte dos métodos e processos relativos a este tema, foi possível identificar os aspetos relevantes a ter em consideração no que toca ao desenvolvimento da solução proposta

A partir dos trabalhos expostos no capítulo 4 é possível verificar que a inserção de publicidade dinâmica e personalizada em *streaming* adaptativo já é um tema de estudo extenso que origina a criação de novos métodos e tecnologias essenciais para a sua evolução. No sentido de satisfazer ambas as necessidades dos consumidores e dos produtores e distribuidores de conteúdo, foram desenvolvidas soluções inovadoras que permitem a inserção de publicidade. No entanto, a publicidade nativa ainda não é contemplada nestes estudos apesar de ser um tema em constante crescimento. Os aspetos recolhidos assim como as análises feitas nos trabalhos aqui referidos são fundamentais para a criação de uma plataforma que pretenda juntar estas funcionalidades.

A solução descrita nesta dissertação serve como ponto de partida para o desenvolvimento de uma plataforma em ambiente *cloud* de distribuição de publicidade nativa com a possibilidade de ser dinâmica, isto é, alternar de campanha publicitária dependendo da localização ou outras características do utilizador que visualiza o conteúdo. É também uma solução que utiliza tecnologias modernas de *streaming* adaptativo de maneira a proporcionar uma melhor experiência ao cliente final. A prova de conceito aqui revelada prova que esta plataforma é possível de ser alcançada através das estratégias desenvolvidas.

A publicidade está a evoluir e a tornar-se cada vez mais sofisticada no que toca à penetração do mercado alvo e à personalização do conteúdo publicitário. A partir da necessidade de contornar os *ad blockers*, a publicidade em vídeo é obrigada a transformar-se e novas plataformas e técnicas estão a ser desenvolvidas para ir ao encontro desta. A solução desenvolvida é capaz de inserir campanhas diferentes de publicidade diretamente na *stream* de vídeo, sobreposta ao conteúdo original, o que impossibilita os *ad blockers* de a removerem sem perderem o conteúdo original. Para além disso, esta abordagem foca-se num tipo de publicidade menos intrusivo e mais subtil que

é mais apelativo às marcas devido ao aumento da desconfiança do público perante a publicidade óbvia e excessiva de produtos.

Tanto para o utilizador final como para os produtores de conteúdo multimédia, esta solução satisfaz as suas necessidades pois não só permite uma boa visualização de conteúdo através do *streaming* adaptativo como também oferece uma nova maneira de gerar publicidade personalizada e campanhas publicitárias eficazes através de uma plataforma eficiente.

7.1 Satisfação dos Objetivos

Grande parte dos objetivos propostos foram concluídos com sucesso e o resultado acrescenta grande valor para a empresa MOG Technologies que poderá utilizar esta solução para o seu projeto e até expandi-la e melhorá-la. No entanto, não foi possível estudar mais ao fundo o desempenho de vários aspetos da solução e por essa razão o objetivo de melhorar e otimizar a solução desenvolvida não ficou concluído.

7.2 Trabalhos Futuros

Apesar de ter sido desenvolvida nesta dissertação uma prova de conceito da solução pretendida, esta ainda não se encontra completamente acabada. Certos aspetos ainda podem ser mais desenvolvidos no sentido de aumentar a robustez da aplicação para ser lançada como um produto viável no mercado. É o caso do armazenamento e a obtenção automática do conteúdo proveniente da empresa Mirriad que não foi o foco deste trabalho, mas que é um aspeto importante para a plataforma.

O estudo de desempenho e funcionalidades de outras tecnologias diferentes para *streaming* adaptativo aplicadas às estratégias desta solução seria um benefício pois tornaria esta mais flexível e consequentemente faria atingir um mercado mais vasto de entidades que utilizam outras tecnologias para além da utilizada neste trabalho.

A comparação dos tempos de execução entre as duas estratégias parciais (A abordagem ESC não foi implementada) referidas nesta dissertação seria um próximo passo para testar qual seria a melhor solução para incorporar a plataforma.

Um tema de estudo interessante para trabalhos futuros seria procurar a otimização do processo de codificação e criação de conteúdo DASH que é o aspeto crítico desta solução. Diferentes perfis do *codec* utilizado ou até *codecs* diferentes poderiam servir como uma solução para reduzir os tempos de execução da plataforma. A tecnologia de *streaming* adaptativo DASH utilizada nesta solução é agnóstica a *codecs* o que torna possível novas abordagens com *codecs* diferentes.

A implementação *cloud* necessita ainda de ser testada com mais rigor e utilizando *clusters* com mais máquinas. Um estudo do comportamento desta solução num ambiente *cloud* mais rigoroso traria mais dados sobre como a aplicação se comportaria num ambiente real que seria bastante útil para a MOG Technologies no que toca à preparação antes do lançamento da plataforma.

Referências

[ABD11] Saamer Akhshabi, Ali C Begen e Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 157–168. ACM, 2011.

[AdA03] AdAge. History: 19th century. Disponível em `http://adage.com/article/adage-encyclopedia/history-19th-century/98706/`, Setembro 2003.

[Ado] Adobe. Http dynamic streaming specification version 3.0 final. Technical report, Adobe.

[AL12] Jens Abrahamsson e Niclas Lindblom. Product placement: A study about swedes attitude towards product placements in movies and tv-shows, 2012.

[Ana16] Enders Analysis. Native advertising in europe to 2020. Disponível em http://www.endersanalysis.com/content/publication/native-advertising-europe-2020, Março 2016.

[Appa] Apple. About http live streaming. https://developer.apple.com/library/content/referencelibrary/Getting

[Appb] Apple. Http live streaming overview. https://developer.apple.com/library/content/documentation/Netw

[ash05] ashleyross. The evolution of advertising: From papyrus to youtube. Disponível em `http://blogs.ubc.ca/etec540sept10/2010/11/29/the-evolution-of-advertising-from-papyrus-to-youtube/`, Novembro 2005.

[Bil15] Ricardo Bilton. Wtf is ad stitching? https://digiday.com/media/wtf-server-side-ad-insertion/, Novembro 2015.

[BKP06] Siva K Balasubramanian, James A Karrh e Hemant Patwardhan. Audience response to product placements: An integrative framework and future research agenda. *Journal of advertising*, 35(3):115–141, 2006.

[CB08] Elizabeth Cowley e Chris Barron. When product placement goes wrong: The effects of program liking and placement prominence. *Journal of Advertising*, 37(1):89–98, 2008.

[DDH+15] Ran Dubin, Amit Dvir, Ofer Hadar, Tomer Frid e Alex Vesker. Novel ad insertion technique for mpeg-dash. In *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, pages 582–587. IEEE, 2015.

[DP99] Bob Dorf Don Peppers, Martha Rogers. Is your company ready for one-to-one marketing. https://hbr.org/1999/01/is-your-company-ready-for-one-to-one-marketing, Janeiro 1999.

REFERÊNCIAS

[Enca] Encoding.com. Global media format report 2016. Technical report, Encoding.com.

[Encb] Encoding.com. Global media format report 2017. Technical report, Encoding.com.

[Encc] Encoding.com. Http dynamic streaming an overview. https://www.encoding.com/http-dynamic-streaming-hds/.

[Encd] Encoding.com. Mpeg-dash an overview. https://www.encoding.com/mpeg-dash/.

[FA00] Rosellina Ferraro e Rosemary J Avery. Brand appearances on prime-time television. *Journal of Current Issues & Research in Advertising*, 22(2):1–15, 2000.

[Fis] Yuval Fisher. An overview of http adaptive streaming protocols for tv everywhere delivery.

[Fuj] Hiroshi Fujisawa. Mpeg-dash and hybridcast. Technical report, Internet Service Systems Research Division.

[HBH17] Bianca Harms, Tammo HA Bijmolt e Janny C Hoekstra. Digital native advertising: practitioner perspectives and a research agenda. *Journal of Interactive Advertising*, pages 1–12, 2017.

[IAB13] IAB. The native advertising playbook. Technical report, IAB, Dezembro 2013.

[Int17] Mordor Intelligence. Video on demand market - by business model (tvod, svod, avod), content (pay tv vod, ott, iptv), application (entertainment, education and training, online commerce, digital libraries), industry, geography, trends, forecast (2017 - 2022). Disponível em https://www.mordorintelligence.com/industry-reports/video-on-demand-market, Novembro 2017.

[Leh07] Jean-Marc Lehu. *Branded entertainment: Product placement & brand strategy in the entertainment business*. Kogan Page Publishers, 2007.

[LWLZ13] Baochun Li, Zhi Wang, Jiangchuan Liu e Wenwu Zhu. Two decades of internet video streaming: A retrospective view. *ACM transactions on multimedia computing, communications, and applications (TOMM)*, 9(1s):33, 2013.

[Med16] MediaRadar. Consumer advertising - maximizing impact. Technical report, MediaRadar, Dezembro 2016.

[Mica] Microsoft. Iis smooth streaming client manifest format. https://msdn.microsoft.com/en-us/library/ee673436(v=vs.90).aspx.

[Micb] Microsoft. Mss introduction. https://msdn.microsoft.com/en-us/library/ff469426.aspx.

[Mor17] M Moroz. Reason, methods and effects of online ad blocking. (863):259–266, 2017.

[Mue15] Christopher Mueller. Mpeg-dash vs. apple hls vs. microsoft smooth streaming vs. adobe hds. https://bitmovin.com/mpeg-dash-vs-apple-hls-vs-microsoft-smooth-streaming-vs-adobe-hds/, Março 2015.

[Net] RGB Networks. Comparing adaptive http technologies. Technical report, RGB Networks.

[Nie16] Nielsen. Video on demand - how worldwide viewing habits are changing in the evolving media landscape. Technical report, Nielsen, Março 2016.

[Ora14] Oracle. Data management platforms demystied, 2014.

[Oze11a] Jan Ozer. What is adaptive streaming? http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-Adaptive-Streaming-75195.aspx, Abril 2011.

[Oze11b] Jan Ozer. What is hls (http live streaming)? http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-HLS-(HTTP-Live-Streaming)-78221.aspx, Outubro 2011.

[PKSA16] Stefan Pham, Christopher Krauss, Daniel Silhavy e Stefan Arbanowski. Personalized dynamic ad insertion with mpeg dash. In *Multimedia and Broadcasting (APMediaCast), 2016 Asia Pacific Conference on*, pages 1–6. IEEE, 2016.

[Pul14] Joe Pulizzi. The ultimate guide to native advertising. Disponível em https://www.linkedin.com/pulse/20140107180859-5853751-the-ultimate-guide-to-native-advertising/, Janeiro 2014.

[Reg17] André Casais Regado. Serviço para product placement em televisão. 2017.

[Sig10] Tim Siglin. Http streaming: What you need to know. http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/HTTP-Streaming-What-You-Need-to-Know-65749.aspx, Fevereiro 2010.

[Sig16] Tim Siglin. A stitch in time: How stream stitching beats the ad blockers. http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/A-Stitch-in-Time-How-Stream-Stitching-Beats-the-Ad-Blockers-108810.aspx, Janeiro 2016.

[Sto11] Thomas Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[Swa13] Viswanathan Swaminathan. Are we in the middle of a video streaming revolution? *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1s):40, 2013.

[TG12] Christian Timmerer e Carsten Griwodz. Dynamic adaptive streaming over http: from content creation to consumption. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1533–1534. ACM, 2012.

[Tuc14] Catherine E Tucker. Social networks, personalized advertising, and privacy controls. *Journal of Marketing Research*, 51(5):546–562, 2014.

[TvR12] Karolina Tutaj e Eva A van Reijmersdal. Effects of online advertising format and persuasion knowledge on audience reactions. *Journal of Marketing Communications*, 18(1):5–18, 2012.

[TVS07] Andrew S Tanenbaum e Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[Vet11] Anthony Vetro. The mpeg-dash standard for multimedia streaming over the internet. Technical report, Mitsubishi Electric Research Labs, 2011.

REFERÊNCIAS

[W3C] W3C. Iso bmff byte stream format. https://www.w3.org/2013/12/byte-stream-format-registry/isobmff-byte-stream-format.html.

[WE16] Bartosz W Wojdynski e Nathaniel J Evans. Going native: Effects of disclosure position and language on the recognition and evaluation of online native advertising. *Journal of Advertising*, 45(2):157–168, 2016.

[WG16] Bartosz W Wojdynski e Guy J Golan. Native advertising and the future of mass communication, 2016.