FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Produção de Televisão *UHD* para Eventos em Direto

Pedro Nuno Vilhena Magalhães

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Dra. Maria Teresa Andrade

28 de Julho de 2016

Resumo

Esta dissertação provém de uma necessidade de conceber e explorar soluções que tornem viável a produção de eventos em direto com uma qualidade bastante superior à atual, em particular recorrendo a conteúdos visuais com Ultra Alta Definição *(UHD)*.

No presente documento descreve-se a implementação de uma solução para produção de televisão que permite a captura do sinal gerado por uma câmara de vídeo de *UHD* recebido via *SDI* e do envio do vídeo capturado sobre *IP*. Esta dissertação foi proposta pela empresa *MOG Technologies* e por isso as soluções concebidas foram integradas num produto da empresa.

Na fase de investigação são apresentados os formatos de vídeo de *UHD*, as fases da produção de televisão e todo o seu *workflow* com foco nas soluções atuais para captura e transporte do vídeo em questão. Segue-se a descrição da implementação das soluções desenvolvidas acompanhada dos detalhes do seu desenvolvimento e explicação das decisões quanto às tecnologias utilizadas e abordagens do estado de arte seguidas.

Por fim, são apresentadas as conclusões e limitações do trabalho desenvolvido bem como um conjunto de boas práticas para que os equipamentos de produção de televisão de *UHD* possam ser implementados com êxito e assim satisfazer as necessidades dos produtores de forma a que este salto tecnológico seja dado com sucesso.

Abstract

This work comes from the need to conceive and explore solutions that make viable the production of live events with superior quality, specifically using visual content with Ultra High Definition (*UHD*).

In this document it's described the implementation of a solution for the production of television that allows the signal capture generated by a *UHD* video camera received via *SDI* and the dispatch of the captured video by *IP*. This dissertation was proposed by *MOG Technologies*, therefore the solutions conceived were integrated in one of the company's products.

In the investigation part are presented the *UHD* video formats, the television production phases and all of its workflow with focus on the actual solutions for the capture and transportation of the respective video. Then, the implementation description of the solutions developed with focus on the details of its development and explanation of the decisions made related to technologies used and states of art approaches used. Finally, there are presented the main conclusions and limitations of the developed work, as well as a good practice set so that the equipments from UHD television production can meet the needs of the producers.

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AMWA	Advanced Media Workflow Association
API	Application Programming Interface
CPU	Central Processing Unit
fps	frames per second
GbE	Gigabit Ethernet
Gb/s	Gigabits Per Second
GUI	Graphical User Interface
HD	High Definition
HBRMT	High Bit Rate Media Transport
Hz	Hertz
ID	Identifier
IP	Internet Protocol
MB/s	Megabytes Per Second
ms	millisegundos
MXF	Material eXchange Format
NMOS	Networked Media Open Specifications
pgroup	pixel group
RAM	Random Access Memory
RFC	Request For Coments
RGB	Red Greend and Blue
RTP	Real-time Transport Protocol
SD	Standard Definition
SDI	Serial Digital Interface
SDK	Software Development Kit
SDP	Session Description Protocol
SMPTE	Society of Motion Picture and Television Engineers
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UHD	Ultra High Definition
UTC	Universal Time Coordinated
UUID	Universally Unique Identifier

Capítulo 1

Introdução

1.1 Contexto

Os eventos em direto são cada vez mais uma das principais fontes de receita das produtoras audiovisuais e das estações de televisão. Esta receita provém maioritariamente dos eventos de âmbito musical e desportivo. Ao mesmo tempo, os formatos de vídeo de Ultra Alta Definição (*UHD*) têm vindo a assumir crescente relevância por todo o mundo, fazendo com que surjam no mercado cada vez mais equipamentos de televisão compatíveis com estes formatos. Assim, os canais televisivos, cientes da importância desta revolução tecnológica, têm vindo a transmitir num caráter ainda exploratório conteúdos nestes formatos.

No entanto, existem ainda várias barreiras que é preciso ultrapassar para se conseguir uma implementação efetiva de sistemas de produção de vídeo *UHD* em tempo real. Um desses desafios está relacionado com a adaptação dos *workflows* de produção e pós-produção do tipo *"file-based"* aos novos formatos. Efetivamente, embora exista já no mercado um número significativo de câmaras *UHD*, o equipamento de estúdio *"file-based"* não está preparado para receber os sinais gerados por essas câmaras.

Para além disso, com o aparecimento dos novos formatos de vídeo, nomeadamente formatos de Ultra Alta Definição, as cadeias convencionais de transporte de sinal de vídeo começam a apresentar limitações que anunciam o seu desaparecimento num futuro próximo. Numa fase de transição, onde os novos formatos ainda não estão completamente desenvolvidos e as plataformas convencionais continuarão a existir, é de extrema importância existir uma plataforma intermédia que permita o ajuste dinâmico aos novos requisitos das novas tecnologias e simultaneamente mantenha a compatibilidade com as plataformas existentes. Essa plataforma foi desenvolvida na segunda fase desta dissertação.

1.2 Motivação

A principal motivação para a realização desta dissertação é a de conceber e explorar soluções que tornem viável a produção de eventos em direto com uma qualidade bastante superior à atual,

em particular recorrendo a conteúdos visuais com Ultra Alta Definição *(UHD)*. A crescente relevância das fontes de receita dos eventos em direto para as estações de televisão e o surgimento dos formatos de vídeo de Ultra Alta Definição tornam este trabalho emergente, atual e precursor.

Embora os formatos de Ultra Alta Definição apresentem um enorme avanço em relação aos formatos de alta definição *(HD)*, o seu aparecimento por ser recente e obrigar a uma remodelação que vai desde a fonte do conteúdo, à cadeia de produção e distribuição, ate à televisão do consumidor final, está a ser explorado num carácter ainda embrionário pelos canais de televisão. Para além disso, a indústria ainda procura as melhores soluções para adaptar estes conteúdos às novas realidades de produção, nomeadamente no que diz respeito ao transporte do vídeo nos formatos de *UHD*. Com os custos associados às redes de *IP (Internet Protocol)* de alta velocidade a descerem continuamente, aliado ao facto de permitirem alta escalabilidade e flexibilidade, esta torna-se uma solução atrativa para esse transporte. Contudo, as ferramentas de transporte de vídeo de Ultra Alta Definição sobre *IP* ainda estão a ser uniformizadas e estudadas para se adaptarem as exigências da atual realidade televisiva, o que torna a sua implementação um desafio.

1.3 Objetivos

Esta dissertação visa, na sua primeira fase, o projeto e implementação de um módulo de *software*, integrado num sistema de captura, que permita a ligação entre o vídeo de *UHD* que sai das câmaras por *SDI* e a restante cadeia de produção do mesmo.

Numa segunda fase, o objetivo é o de desenvolver um módulo que possibilite a transmissão do vídeo capturado sobre uma rede *IP* utilizando o *Real-Time Protocol (RTP).*

Esta dissertação, visa também a definição de um conjunto de regras e boas práticas que permitam aos produtores de televisão encarar este salto tecnológico com a confiança necessária e criar as condições para que seja possível produzir eventos em direto com uma qualidade bastante superior à atual.

O trabalho foi realizado na empresa *MOG Technologies* e por isso os módulos a desenvolver destinam-se a ser integrados num sistema de captura e *ingest* da empresa.

1.4 Estrutura da Dissertação

Este documento foi dividida em cinco capítulos principais. No primeiro e presente capítulo é apresentada uma introdução ao âmbito da dissertação explicando o contexto, as motivações e os objetivos da mesma. O segundo capítulo diz respeito ao estudo do estado da arte onde é feita uma abordagem aos conceitos associados à dissertação e ao trabalho já desenvolvido neste âmbito. De seguida, no terceiro capítulo, é explicada a implementação da solução desenvolvida bem como as decisões tomadas nas diferentes fases da implementação. No quarto capítulo são apresentados os resultados obtidos da implementação e é feita uma avaliação dos mesmos. Por fim, no quinto capítulo são apresentadas as principais conclusões do trabalho desenvolvido bem como possíveis trabalhos futuros que podem resultar desta dissertação.

Capítulo 2

Revisão Bibliográfica

Neste capítulo começam-se por descrever as fases da Cadeia de Valor da televisão de forma a apresentar uma visão geral do âmbito da dissertação. De seguida, é abordada com mais detalhe a fase particular da cadeia de valor da televisão onde a dissertação se insere, a fase de Produção de Televisão. Neste subcapítulo são analisadas todas as etapas inerentes à produção de televisão assim como as diferentes infraestruturas para produção existentes.

Seguidamente, são apresentados os conceitos de *RAW* e vídeo não comprimido. Estes conceitos são muitas vezes confundidos e um esclarecimento dos mesmos é necessário para compreender o trabalho desenvolvido nesta dissertação.

No subcapítulo seguinte, *Ultra High Definition Television*, são explicados em detalhe os formatos de Ultra Alta Definição utilizados em televisão.

De seguida, são apresentadas as soluções de transporte de vídeo de *UHD* relevantes para a produção de televisão: as soluções de Transporte de vídeo sobre *SDI*, que já são largamente adotadas pela indústria desde há bastante tempo e as soluções mais recentes de Transporte de vídeo sobre *IP*.

2.1 Cadeia de Valor da televisão

A cadeia de valor da televisão pode ser vista como compreendendo quatro fases sequenciais: Produção do conteúdo, Programação, Distribuição e Entrega e Consumo [12]. Em resumo, um conteúdo de televisão é produzido, armazenado, inserido numa grelha de programação e enviado para para os *broadcasters* que distribuem esse conteúdo para o consumidor final.

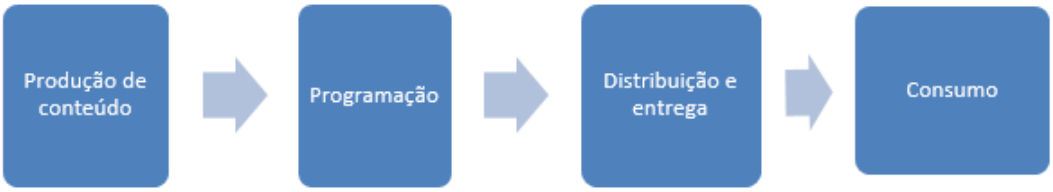

Figura 2.1: Fases da cadeia de valor da televisão.

A cadeia inicia-se assim com a fase de produção de conteúdo que está por sua vez dividida em três partes: **pré-produção, produção e pós produção** [13].

A pré-produção engloba a idealização do conteúdo, a escrita do guião, a identificação e caracterização do local de captura, incluindo a gestão da equipa de produção, atores, decoração dos cenários, etc. Após a idealização e a aprovação do conteúdo segue-se a fase de produção propriamente dita que ocorre desde o momento em que a equipa se dirige para o local de gravação até à captura ou transmissão de um determinado conteúdo ou evento para o local da pós-produção. Este tipo de transmissão dos sinais audiovisuais é designado de "contribuição" e normalmente este termo é utilizado sempre que se transferem conteúdos entre equipamento audiovisual e centros de produção onde vai ocorrer a pós-produção. Associa-se a este tipo de transmissão uma qualidade dos conteúdos superior à qualidade com que os conteúdos são difundidos para o telespectador. O mesmo acontece com a designada "distribuição primária", termo utilizado para designar o transporte de sinal para centros de distribuição que podem ou não incluir pós-produção [14]. Nesta fase estão incluídas as etapas de captura e *ingest*. É na fase de produção que se enquadra esta dissertação.

Depois de captados e armazenados, o vídeo e o áudio passam por uma série de etapas responsáveis pelo seu processamento e edição que constitutem a fase de pós-produção. Aqui, o conteúdo é recolhido, preparado, editado e finalizado com vista à sua transmissão *(broadcasting)* e/ou arquivo.

A fase que se segue à pós-produção é a fase da programação protagonizada pelos fornecedores/distribuidores de conteúdo. Nessa fase os conteúdos são inseridos numa grelha de programação, podendo ser feita inserção a inserção de publicidade. No caso de conteúdos que são transmitidos em diferido é ainda criada uma cópia do conteúdo para arquivo. Os conteúdos já inseridos na grelha de programação e eventualmente intercalados com publicidade encontram-se prontos para serem distribuídos. No caso dos eventos em direto, como é o caso que esta dissertação aborda, não ocorre armazenamento em arquivo antes da distribuição e entrega. A fase seguinte abrange pois a distribuição dos conteúdos por difusão até o cliente final. Assim, resta apenas ao cliente usufruir do conteúdo: fase de consumo.

2.2 Produção de Televisão

Por ser a fase da cadeia de valor da televisão em que se enquadra esta dissertação, neste capitulo será analisado o processo de produção de televisão de uma forma mais pormenorizada.

Durante muitos anos, o suporte mais utilizado em todo o processo de produção de televisão era a fita magnética estando os conteúdos gravados em cassetes *(tapes)* [13]. Estes sistemas baseados em cassete, sistemas *tape-based*, permitem a gravação de vídeo/áudio analógico que pode acontecer diretamente para a cassete presente na câmara ou através de um gravador externo, como um *VTR (Video Tape Recorder*). Depois da gravação, é necessário identificar manualmente as cassetes, descrever o seu conteúdo e transportá-las até à fase seguinte da infraestrutura onde se inserem. Como as cassetes existem em várias formas e tamanhos deve existir um cuidado adicional na escolha do equipamento de todo o *workflow* para não existirem problemas de interoperabilidade.

As cassetes, por serem um equipamento físico e analógico têm um tempo de vida limitado que varia obviamente com a frequência de utilização das mesmas. Também por causa disso, as fitas magnéticas permitem apenas acesso sequencial, pelo que para efeitos de edição do conteúdo é necessário percorrer a fita *(linear editing)* e para unir várias fitas, é necessário *hardware* específico. Todo este processo é demorado e trabalhoso.

Um exemplo do *workflow* de produção de um sistema *tape-based* é apresentado na Figura 2.2.

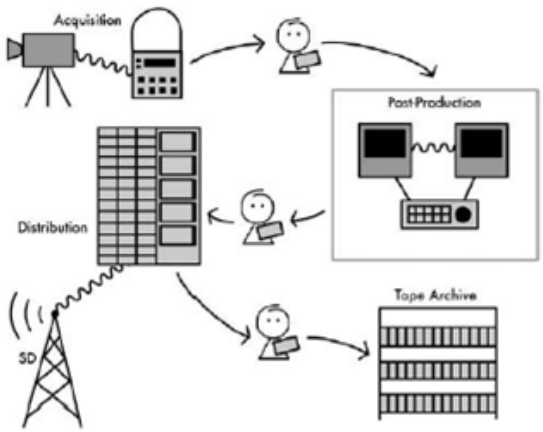

Figura 2.2: *Workflow* de uma infraestrutura de produção televisão baseada em cassetes/fita.[1]

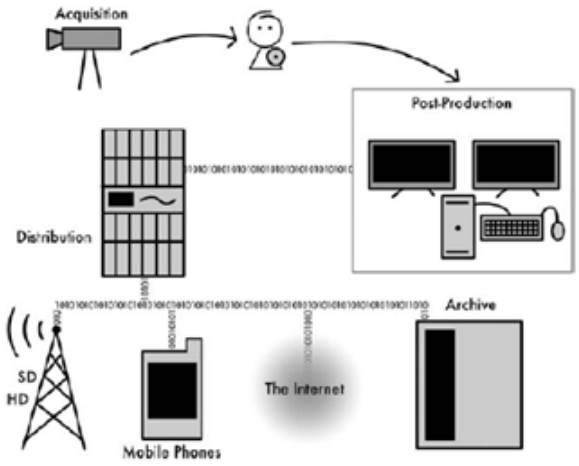

Figura 2.3: *Workflow* de uma infraestrutura de produção televisão baseada em ficheiros.[1]

As infra-estruturas de produção de televisão baseadas em sistemas que utilizam cassete/fita *(tape-based)* apresentam grandes exigências de intervenção humana quer para o transporte quer para a catalogação das cassetes e descrição dos seus conteúdos. Apresentam ainda limitações no acesso aleatório e na realização de operações de edição, para além de não oferecerem garantias de interoperabilidade entre sistemas e de trazerem grande complexidade à manutenção do arquivo.

Assim, em paralelo com a introdução da adesão em massa às tecnologias digitais, as infraestruturas baseadas em cassetes começaram a ser substituídas por infraestruturas baseadas em ficheiros *(file-based)* . Estas infraestruturas eliminam consideravelmente a necessidade de intervenção humana, os ficheiros são digitais e por isso armazenados e acedidos com muito mais facilidade. Permitem o acesso não sequencial aos conteúdos e a sua edição não destrutiva na medida em que através de um editor digital é possível criar várias versões do mesmo conteúdo. As infraestruturas *file-based* são também mais flexíveis na medida em que dependem menos de *hardware* e mais de *software*, adaptando-se a novas exigências tecnológicas mais facilmente. Este tipo de infraestruturas é o considerado no âmbito desta dissertação. O *workflow* de uma infraestrutura de produção de televisão baseada em ficheiros é apresentada na Figura 6. No caso dos eventos em direto, a intervenção humana seria forçosamente eliminada entre a aquisição ("acquisition") e a pós-produção ("pos-production"), podendo o conteúdo ser enviado entre estas etapas, por exemplo, por *SDI* ou *IP*.

2.2.1 Ambientes Profissionais *File-based* de Produção de Televisão

Atualmente um ambiente profissional *file-based* de produção de qualquer conteúdo multimédia, no caso particular desta dissertação, de produção de televisão, possui um *workflow* associado dividido em diversas fases. Estas fases são ilustradas na Figura 2.4. A captura e o *ingest* fazem parte da etapa de produção, à qual se seguem as etapas de pós-produção e transmissão. [15]

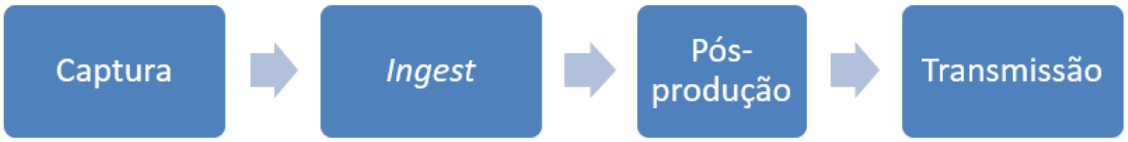

Figura 2.4: *Workflow* de um ambiente profissional de produção de televisão

2.2.1.1 Captura

A primeira fase da produção de televisão é a captura do conteúdo emitido por uma fonte de sinal de vídeo, normalmente uma câmara, e áudio, normalmente microfones. Esta fase é crucial para o sucesso da produção de conteúdo pois por ser o início da cadeia, qualquer erro vai ser difícil ou mesmo impossível de resolver em todas as fases seguintes da cadeia de televisão.

A captura consiste então na aquisição do sinal emitido por uma ou mais câmaras e na transcodificação desses dados para o formato de input necessário pelo equipamento da fase seguinte, a fase de *ingest*. O sinal normalmente não sofre compressão para que se possa obter a melhor qualidade possível nesta fase inicial e também para efeitos de interoperabilidade com a restante cadeia de produção. Este, é geralmente *RAW* ou um formato de vídeo não comprimido e a diferença entre os dois é descrita em detalhe no subcapítulo 2.3.

Atualmente na fase de captura assim como em toda a cadeia de produção de televisão, o sinal é na maioria das situações transportado através de *SDI* (*Serial Digital Interface)*, este *interface* é descrito com mais detalhe no subcapítulo 2.5.1 deste documento.

2.2.1.2 *Ingest*

Ingest é o processo em que o vídeo capturado é encapsulado num formato específico juntamente com outros dados como áudio ou informação auxiliar para ser armazenada e/ou distribuída para a fase seguinte da cadeia de produção onde se insere. [16]

O sinal visual pode ser capturado em diferentes formatos, adotando distintas normas ou utilizando diferentes parâmetros e configurações, cuja utilização é muita vezes ditada por requisitos da aplicação em vista (fim a que os conteúdos se destinam) mas também por limitações do equipamento de captura. Por esse motivo, o equipamento que é utilizado nas operações de pós-produção deve ser suficientemente flexível para poder aceitar, converter e manipular diferentes formatos e tipos de ficheiros. Assim, será capaz de lidar quer com as limitações do equipamento de captura quer com a variedade de requisitos impostos pela aplicação, a que os conteúdos se destinam, nomeadamente edição ou armazenamento.

De seguida, é necessário juntar o conteúdo de vídeo, com o conteúdo áudio correspondente, assim como alguns metadados necessários aos mesmos, que podem ser por exemplo, descrições desses ficheiros, legendas, identificadores, etc. Esta operação, chamada *wrapping*, consiste na prática em encapsular num único ficheiro, diferentes tipos de dados relacionados com o mesmo

conteúdo (vídeo, áudio e metadados), sendo também responsável por garantir que estes se mantêm sincronizados [17]. O formato *MXF* é o formato resultante desta operação mais utilizado na indústria da televisão.

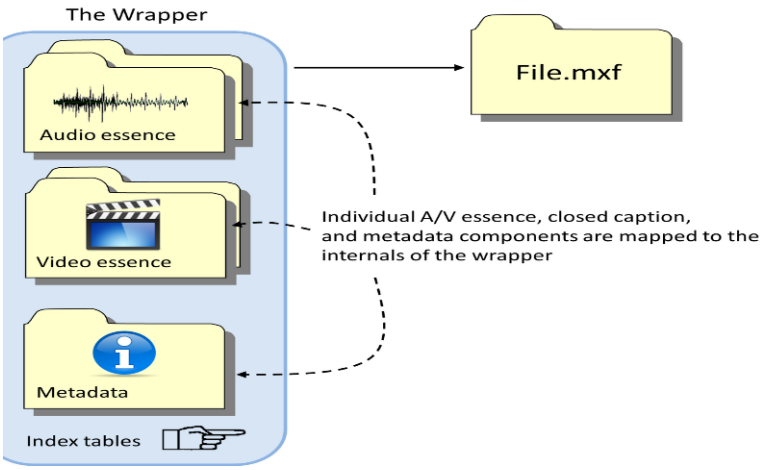

Figura 2.5: Ilustração do conteúdo de um ficheiro *MXF*.[2]

2.2.1.3 Pós-produção e Transmissão

Segue-se a fase de pós produção, onde o conteúdo audiovisual proveniente da fase de *ingest* é editada. Hoje em dia a edição é conseguida através de software como por exemplo o *Final Cut Pro* da *Apple* ou o *Adobe Premiere* da *Adobe*. As operações realizadas nesta etapa são variadas e de naturezas diferentes, tal como ajustes das característica de imagem ou áudio, inserção de efeitos ou outras imagens e vídeos no conteúdo final, entre outros.

Após a fase de pós-produção, os conteúdos estão prontos para serem transmitidos.

2.3 *RAW* e vídeo não comprimido

Embora o termo *RAW* se refira aos dados que representam toda a informação referente ao sinal vídeo capturado é diferente de vídeo não comprimido. Vídeo não comprimido é o formato a ser utilizado no âmbito desta dissertação pelo que neste capítulo serão apresentadas as diferenças entre os dois.

Para entender o que é *RAW*, é necessário entender o processo de captura de vídeo digital por uma câmara. O que acontece em primeiro lugar é a reflexão da luz nos sensores da câmara. Esses sensores são monocromáticos e são apenas sensíveis à intensidade luminosa, sendo a informação da cor de cada píxel obtida através de um filtro de cor que define para cada pixel um único valor de vermelho, verde ou azul. Os ficheiros que contêm as imagens resultantes desta filtragem monocromática denominam-se *RAW*. [2]

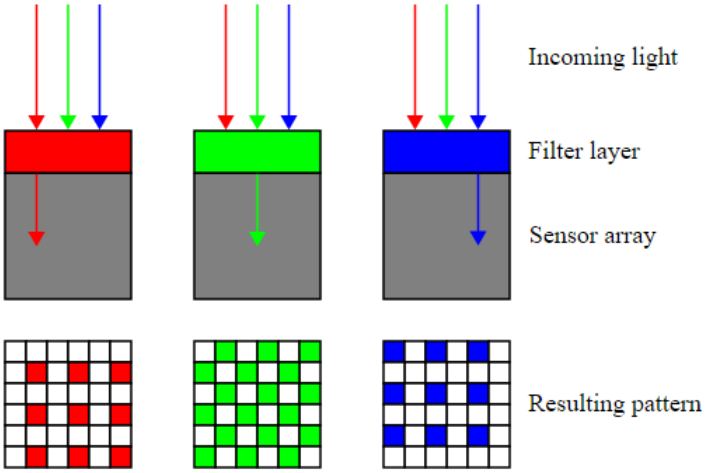

Figura 2.6: Esquema exemplo do processo de reflexão de luz, filtragem de cor pelo sensor de uma câmara e padrão resultante. [3]

Como cada píxel contém apenas um valor de uma destas três cores, um ficheiro *RAW* não é visível de forma viável num monitor pois para esse efeito cada um necessita de informação completa de cor.

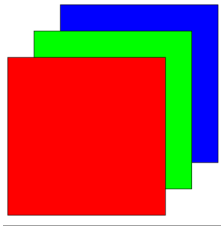

Figura 2.7: As cores primárias *"Red, Green and Blue"(RGB)* necessárias para definir a cor "real"de um píxel. [3]

Assim, para poderem ser visualizados de forma mais realista os dados *RAW* necessitam de ser processados e interpretados através da interpolação dos diferentes pixéis de cada imagem para que se consiga obter a informação completa de cor para cada um.

Figura 2.8: Esquema de obtenção da cor de um píxel digital utilizando o padrão *RGB*. A cor final é obtida somando as componentes de cada cor primária. [3]

Esta conversão acontece na maior parte das vezes na própria câmara, embora algumas câmaras forneçam também o ficheiro *RAW* para que o usuário utilize um *software* ou *hardware* de uma terceira entidade que a implemente. Como essa conversão pode consumir muito tempo no processo de pós-produção, em muitos âmbitos opta-se por obter diretamente da câmara o resultado dessa conversão.

Cada fabricante de câmaras tem o seu próprio sistema e por isso um conversor *RAW* deve ter em conta as características de cada sensor. Idealmente as imagens resultantes seriam perfeitas: não comprimidas, com cores bem definidas, correspondendo à "imagem real" na perfeição. No entanto, isso não se verifica pois os conversores *RAW* necessitam de ter em conta várias variáveis como o espaço de cor utilizado pelo sensor, definições *gamma*[1] , qual o tipo de *alisaing*[2] presente na imagem e a causa do mesmo, como a amostragem escolhida afeta a resolução, contraste entre cores, etc. Assim, a imagem final, é o resultado de um conjunto de várias instruções sobre como interpretar a luz que foi refletida no sensor, sob certas condições, ajustadas às capacidades da câmara. A essas imagens combinadas e apresentadas em intervalos de tempo bem definidos, dá-se o nome de **vídeo não comprimido.**[3]

O termo vídeo não comprimido, introduz a ideia de que não existe qualquer tipo de compressão relativo ao vídeo capturado, mas na realidade, na maior parte das vezes isto não se verifica. Como referido anteriormente, o vídeo resulta de dados *RAW* que contém informação de píxeis monocromáticos que são representados com uma profundidade de cor entre 12 e 16 *bits*. Esse valor para o vídeo não comprimido está limitado geralmente entre 8 a 12 *bits*. Esta limitação, deve-se ao facto de cada pixel necessitar de informação completa de cor e brilho (*RGB* ou *YUV*) o que implicaria que cada pixel de vídeo não comprimido fosse representado por 3 vezes mais *bits* do que um pixel em *RAW*, tornando o tamanho do vídeo resultante num ficheiro muito grande. Para além da redução da profundidade de cor , também a informação de cor é geralmente reduzida de *4:4:4* para *4:4:2*. Concluindo, o vídeo não comprimido, embora não sofra compressão espacial ou temporal sofre na maioria das vezes subamostragem de cor e redução da profundidade de cor. [3]

2.4 *Ultra High Definition Television*

Assim como acontece com o nome e acrónimos de várias tecnologias, existem diversas interpretações e definições que se vulgarizam referentes ao mesmo termo, é o caso do termo *Ultra High Definition Television (UHDTV)*, em português "Televisão De Ultra Alta Definição". O termo "Ultra" é normalmente associado a uma nova tecnologia que indica um *upgrade* ao seu antecessor e por isso nos formatos de televisão apareceram diversos conceitos como "4k" e "8k" que nem sempre correspondem à definição científica mais correta.

O termo mais formal de Televisão de Ulta Alta Definição (*UHDTV*) foi apresentado pelo *NHK Science & Technology Research Laboratories* e aprovado e normalizado pela *ITU (International Telecommunication Union)* sobre a *ITU-R Recommendation BT.2020*, vulgarmente designada por

[1] Relação entre o valor numérico de um píxel e a sua luminância.
[2] Efeito corrosivo na imagem devido à amostragem do sinal.

Rec. 2020 ou BT. 2020 [18] , e pelo *SMPTE (Society of Motion Picture and Television Engineers)* sobre a norma *SMPTE ST 2036* [19]. Essas normas definem dois formatos, o *UHDTV-1* (3840x2160 píxeis), referido também como *UHDTV1*, e o U*HDTV-2* (7680x4320 pixéis), também referido como *UHDTV2*, e incluem a definição de vários parâmetros dos formatos como a resolução temporal e espacial da imagem, espaço de cor, profundidade de cor. A norma *SMPTE ST 2036* define ainda a estrutura do áudio, interface digital e mapeamento do vídeo.

As diferentes partes da norma *SMPTE 2036* são: [19]

- *SMPTE ST 2036-1: "Ultra High Definition Television – Image Parameter Values for Program Production;*
- *SMPTE ST 2036-2; "Ultra High Definition Television- Audio Charachteristics and Audi Channel Mapping for Program Production";*
- *SMPTE ST 2036-3: "Ultra High Definition Television- Mapping into Single-link or Multi-link 10 Gb/s Serial Signal/Data Interface";*
- *SMPTE ST 2036-4, "Ultra High Definition Television- Multi-link 10 Gb/s Signal/Data Interface Using 12-Bit Width Container*

A Figura 2.9 ilustra a diferença em termos de resolução espacial entre os dois formatos de *UHDTV* e os seus antecessores *HD (High definition)* e *SD (Standard Definition)*.

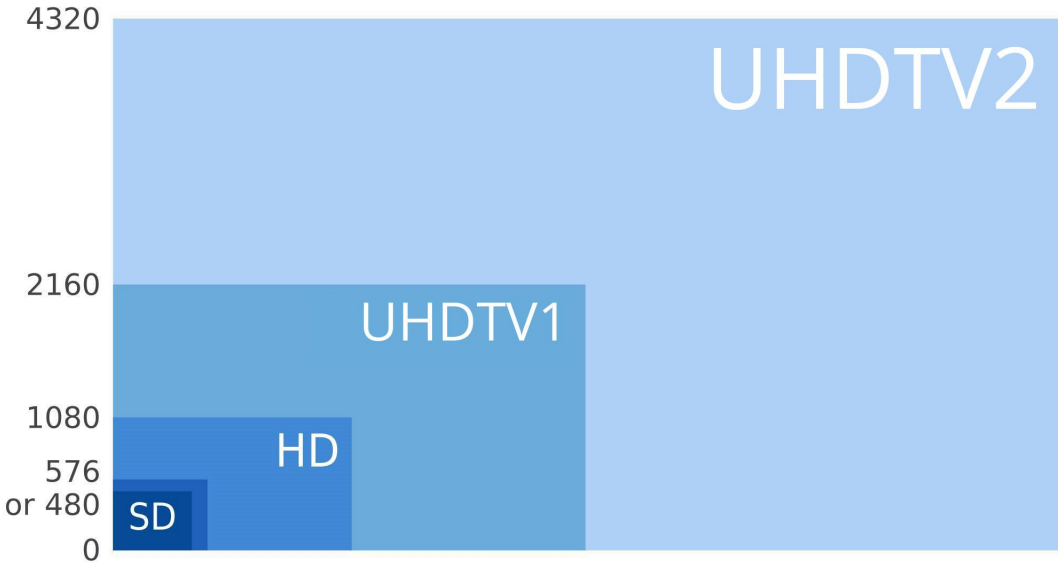

Figura 2.9: Ilustração da resolução espacial dos formatos *SD, HD, UHDTV1* e *UHDTV2*.

2.4.1 Caracterização da imagem

As características de imagem dos formatos *UHDTV1* e *UHDTV2* são apresentadas de forma resumida na Figura 2.10.

System category	System nomenclature	Luma or R' G' B' samples per line	Lines per frame	Frame rate (Hz)
UHDTV1	3840 × 2160/23.98/P	3840	2160	24/1.001
	3840 × 2160/24/P	3840	2160	24
	3840 × 2160/25/P	3840	2160	25
	3840 × 2160/29.97/P	3840	2160	30/1.001
	3840 × 2160/30/P	3840	2160	30
	3840 × 2160/50/P	3840	2160	50
	3840 × 2160/59.94/P	3840	2160	60/1.001
	3840 × 2160/60/P	3840	2160	60
	3840 × 2160/120/P	3840	2160	120
UHDTV2	7680 × 4320/23.98/P	7680	4320	24/1.001
	7680 × 4320/24/P	7680	4320	24
	7680 × 4320/25/P	7680	4320	25
	7680 × 4320/29.97/P	7680	4320	30/1.001
	7680 × 4320/30/P	7680	4320	30
	7680 × 4320/50/P	7680	4320	50
	7680 × 4320/59.94/P	7680	4320	60/1.001
	7680 × 4320/60/P	7680	4320	60
	7680 × 4320/120/P	7680	4320	120

Figura 2.10: Estrutura da imagem e *frame rate* dos formatos de *UHDTV*. [4]

Na coluna *"System nomenclature"* são apresentados o número de píxeis em cada linha, o número de píxeis em cada coluna, o *frame rate* e que o varrimento da imagem é progressivo para todos os formatos de *UHD*, indicado pela letra *"P"*.

Os píxeis de cada imagem são uniformemente espaçados, ortogonais, quadrados e apresentam um *aspect ratio* de *1:1* e por isso o *aspect ratio* de cada *frame* de ambos os formatos é de *16:9*. A ordenação dos píxeis em cada linha é feita da esquerda para a direita e as linhas são ordenadas de cima para baixo.

2.4.2 Colorimetria

Os valores de referência das cores primárias e branco estão especificados na norma *SMPTE 2036-1*, estes valores estão definidos entre 0 e 1 e de acordo com o espaço de cor *CIE 1931 XY* [18]. Este espaço de cor, criado pela *International Comissiono n Illumination* em 1931, define as relações quantitativas entre as cores físicas "puras" do espetro visível (definidas pelo seu comprimento de onda) e a perceção de cor fisiológica humana, englobando todas as cores percetíveis ao ser humano [20].

	CIE x	CIE y
Red primary	0.708	0.292
Green primary	0.170	0.797
Blue primary	0.131	0.046
Reference white	0.3127	0.3290

Figura 2.11: Cores primárias e branco de referência para *UHDTV*. [4]

Para fins de compatibilidade com o restante equipamento dos diversos *workflows* de *HDTV (“backward compatibility”)* , a norma *ST 2036-1* ainda permite que se adote outros valores de referência para as cores primárias e branco no âmbito da *UHDTV1*. Estes valores são os mesmos usados na norma *ITU BT 709* referente a *HDTV* e são apresentados na figura seguinte.

	CIE x	CIE y
Red primary	0.640	0.330
Green primary	0.300	0.600
Blue primary	0.150	0.060
Reference white	0.3127	0.3290

Figura 2.12: Cores primárias e branco de referência alternativos para *UHDTV*. [4]

O espaço de cor das duas abordagens referidas em cima são apresentados na Figura 2.13. Estes estão delimitado pelos triângulos a preto no caso dos valores de referência do *ST2036-1* e *ITU-R BT.202* e a cinzento no caso da abordagem opcional concordante com o *ITU-R BT.709*. Este espaço é caracterizado por um parâmetro de luminância *"Y"* e duas coordenadas espaciais x e y que especificam um ponto no diagrama de cromaticidade. No primeiro caso (*SMPTE 2036-1/ITU-R BT 20220*) é coberto 75,8% do espaço de cor *CIE 1931* enquanto que no segundo caso *(ITU-R BT.709)* é coberto 35,9% desse espaço. [4]

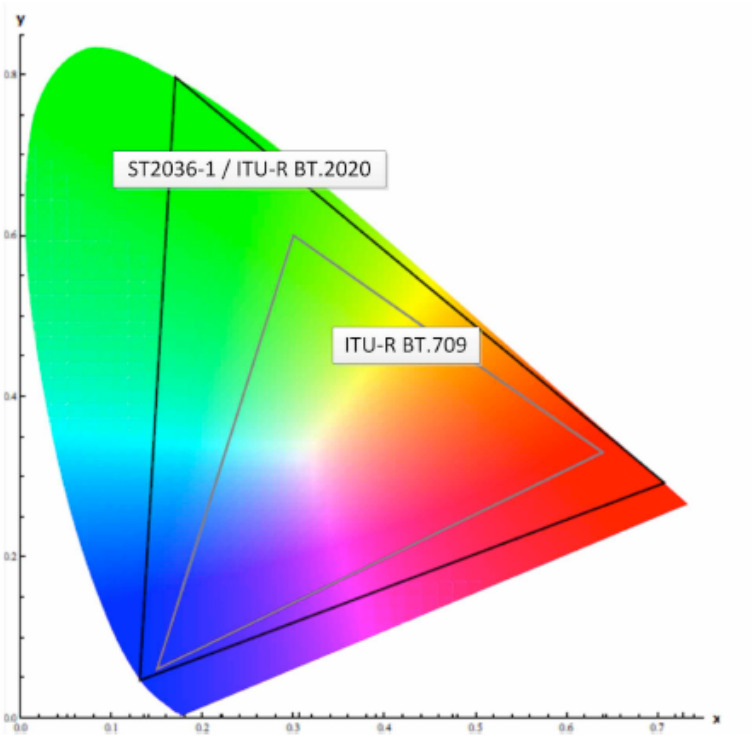

Figura 2.13: Espaço de cor para *UHDTV*. [4]

2.4.3 Representação Digital do Sinal

Nas seguintes tabelas são apresentados os payloads de vídeo não comprimido para os formatos *UHDTV-1* e *UHDTV-2* respetivamente.

Horizontal Pixels	Vertical Pixels	Frames per Second (nominal)	Total Payload (nominal)					
			10-bit 4:2:0	10-bit 4:2:2	10-bit 4:4:4	12-bit 4:2:0	12 Bit 4:2:2	12-bit 4:4:4
3840	2160	120	15Gbit/s	20Gbit/s	30Gbit/s	18Gbit/s	23Gbit/s	36Gbit/s
		60	7.5Gbit/s	10Gbit/s	15Gbit/s	9Gbit/s	12Gbit/s	18Gbit/s
		50	6Gbit/s	8Gbit/s	12Gbit/s	7.5Gbit/s	10Gbit/s	15Gbit/s
		30	3.7Gbit/s	5Gbit/s	7.5Gbit/s	4.5Gbit/s	6Gbit/s	9Gbit/s
		25	3.1Gbit/s	4 Gbit/s	6.2Gbit/s	3.7Gbit/s	5Gbit/s	7.5Gbit/s
		24	3Gbit/s	4Gbit/s	6Gbit/s	3.6Gbit/s	4.8Gbit/s	7.2Gbit/s

Figura 2.14: *Payload* de vídeo *UHDTV1* não comprimido. [4]

Horizontal Pixels	Vertical Pixels	Frames per Second (nominal)	Total Payload (nominal)					
			10-bit 4:2:0	10-bit 4:2:2	10-bit 4:4:4	12-bit 4:2:0	12-bit 4:2:2	12-bit 4:4:4
7680	4320	120	60Gbit/s	80Gbit/s	120Gbit/s	72Gbit/s	95.5Gbit/s	144Gbit/s
		60	30Gbit/s	40Gb/s	60Gbit/s	36Gbit/s	48Gbit/s	72Gbit/s
		50	25Gbit/s	33Gbit/s	50Gbit/s	30Gbit/s	40Gbit/s	60Gbit/s
		30	15Gbit/s	20Gbit/s	30Gbit/s	18Gbit/s	24Gbit/s	36Gbit/s
		25	12.4Gbit/s	16,6Gbit/s	25Gbit/s	15Gbit/s	20Gbit/s	30Gbit/s
		24	12Gbit/s	16Gb/s	24Gbit/s	14.4Gbit/s	19 Gbit/s	29Gbit/s

Figura 2.15: *Payload* de vídeo *UHDTV2* não comprimido. [4]

Os sistemas de *UHDTV* são progressivamente capturados a *frame rates* de *24/1.001 fps; 24 fps; 25 fps; 30/1.001 fps; 30 fps; 50 fps; 60/1.001 fps; 60 fps; or 120 fps.* Cada píxel é codificado com 10 ou 12 *bits (bit depth)* e a sub-amostragem de cor pode ser *4:4:4, 4:2:2 ou 4:2:0*. O tamanho do *payload* total de *UHDTV* é significativamente mais alto do que o dos formatos anteriores devido ao número de pixéis horizontais e píxeis verticais e também porque é utilizado exclusivamente varrimento progressivo. Estes são os primeiro formatos em que não é definida a opção de varrimento entrelaçado, que seria capaz de alterar o tamanho do *payload* para metade.

2.5 Transporte de vídeo sobre *SDI*

2.5.1 Serial Digital Interface

Serial Digital Interface (SDI) é uma famíla de normas criada pela *SMPTE* para transmissão de vídeo digital não comprimido através de cabo coaxial ou fibra ótica, utilizado na grande maioria dos equipamentos profissionais de televisão.

Esta especificação destina-se ao transporte não só do sinal de vídeo, mas também o sinal de áudio, o *timecode* da transmissão que permite o registo temporal do vídeo para efeitos de sincronização, assim como outra informação auxiliar (metadados). A este conjunto de dados transportados para além do vídeo, chama-se *ancilliary data* (abreviada para *ANC data*), conceito introduzido pela *SMPTE* em 1998 [21]. Estes dados são incluídos no sinal *SDI*, ao lado e/ou em cima da região onde é transportada a imagem.

Na indústria de *broadcast* o transporte de vídeo tem vindo a ser quase exclusivamente feito sobre *SDI* devido a vários fatores tal como a existência de várias normas definidas e largamente adotadas pela indústria, conexão simples do tipo *"point-to-point plug and play*", grande disponibilidade, baixa latência, entre outros. [20]

Esta *interface*, por ser exclusiva da índustria de televisão, adapta-se a esta de acordo com as suas necessidades. Quando surgem novos formatos, como é o caso dos formatos de ultra alta definição, é preciso criar soluções que possibilitem o seu transporte da melhor forma possível.

2.5.2 Transporte de vídeo *UHD* sobre *SDI*

Como foi referido em 2.4.3 e exposto nas Figuras 2.14 e 2.15, os *payloads* dos formatos de ultra alta definição requerem uma grande largura de banda. Quando comparada à largura de banda do formato antecessor , o formato de alta definição *(HD)*, o formato de *UHDTV* com menos píxeis, o *UHDTV1*, requer uma largura de banda superior em cerca de oito vezes. Para ser transmitida com qualidade fiável é necessária uma interface que suporte pelo menos 12 *Gb/s* de *bitrate* [5]. Ora, este valor ultrapassa largamente o valor de 1.485 *Gb/s* suportado pelo *SDI* desenvolvido para *HDTV*, o *HD-SD* especificado na norma *SMPTE 372M*, em 1998). O transporte de *UHDTV1* será analisado no resto do capítulo, por ser o formato que se enquadra no âmbito desta dissertação uma vez que a placa de captura de vídeo sobre *SDI* utilizada só suporta até este formato, como será explicado em 3.3.3.

A primeira solução adotada para o transporte do formato de ultra alta definição *UHDTV1* foi conseguida através da divisão de uma imagem *UHDTV1* em quatro quadrantes *HD*, transportando-os em quatro sinais *HD-SDI* sincronizados. Esta solução permite o transporte de vídeo *UHD* até 30 *fps* por quatro sinais *HD-SDI*, ou até 60 *fps* através de quatro sinais *3G-SDI* [5].

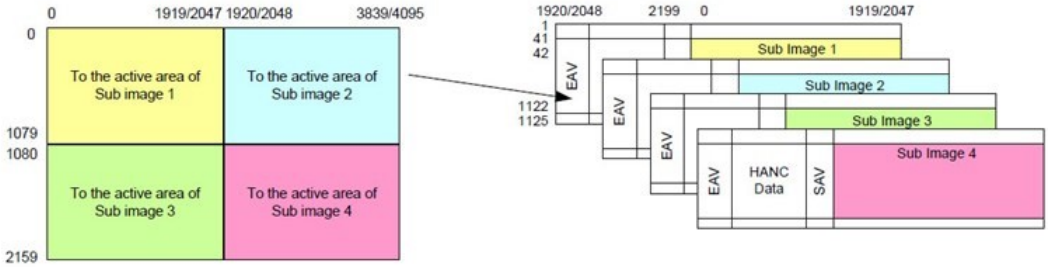

Figura 2.16: Divisão de uma imagem *UHD* em 4 sub-imagens *HD* [5]

Em 2013 e em 2015 a *SMPTE* definiu as normas *425-3* e *425-5* respetivamente. Estas normas definem um novo método de mapeamento de imagens *UHDTV1* com débitos de até 10 Gb/s em diferentes imagens *HD*, transportadas em 2 (*dual-link*) e 4 (*quad-link*) cabos *3G-SDI* respetivamente. Ao contrário da solução anterior, onde a imagem *UHDTV1* era dividida em quatro regiões e cada uma correspondia a uma sub-imagem *HD*, nesta nova abordagem cada sub-imagem é obtida dividindo a imagem *UHDTV1* em vários quadrados de quatro *bits*, sendo que cada linha desse quadrado corresponde a dois bits de uma sub-imagem. Um exemplo ilustrativo deste mapeamento segundo a norma *SMPTE 425-5* é apresentado na figura 2.17.

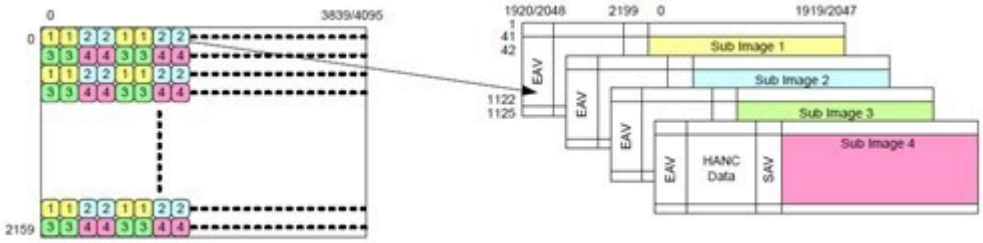

Figura 2.17: Mapeamento de uma imagem *UHD* em 4 subimagens *HD* segundo a norma *SMPTE 425-5* [5]

Com o aparecimento do *6G-SDI* e *12G-SDI* foram definidos novos *standards* que permitem o transporte de vídeo com *payloads* maiores. O *SMPTE ST 2081*, definido em 2015, define o transporte de vídeo com *payload* até 20 *Gb/s* por cabos *6G-SDI* em *single-link*, *dual-link* e *quad-link*. A norma *SMPTE ST 2082*, de 2015 define o transporte de vídeo com *payloads* até 40 *Gb/s* por cabos *6G-SDI* em *single-link*, *dual-link* e *quad-link*.

Figura 2.18: Ilustração de diferentes tipos de transporte de um vídeo com *payload* até 12 *Gb/s*. Da esquerda para a direita: *quad-link 3G-SDI*, *dual-link 6G-SDI* e *single-link 12G-SDI*.[5]

As possibilidades para o transporte de vídeo com *payload* até 12 *Gb/s* por *SDI* estão resumidas pelo esquema da *Figura 20*. Começa-se assim por dividir o *frame* da fonte *("Source Image")* em quatro sub-imagens que contêm partes desse *frame*. Segue-se o mapeamento *("mapping")* das diferentes imagens de acordo com o formato *SDI* a utilizar e um passo intermédio, que pode ser necessário ou não, chamado *"Gearbox Remapping"* e que antecede o envio do vídeo para outra infraestrutura da cadeia em que se insere. Este passo diz respeito aos casos em que se pretende enviar vídeo num formato *SDI* diferente do que foi utilizado no mapeamento.

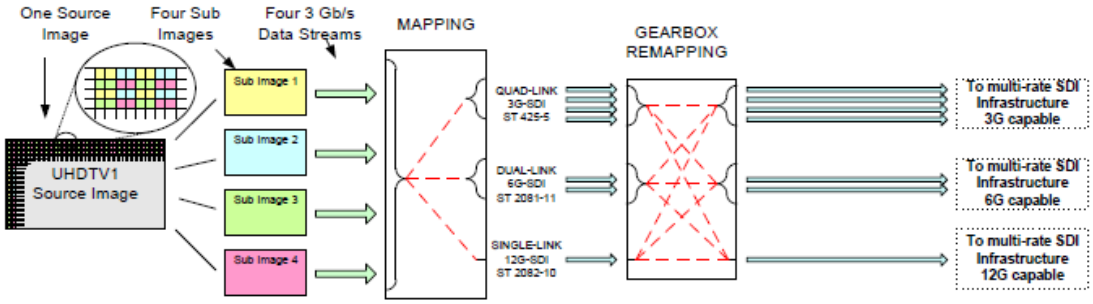

Figura 2.19: Exemplo de um processo de envio de um vídeo de *UHD* com *payload* até 12 *Gb/s* sobre *SDI* [6]

2.6 Transporte de vídeo sobre *IP*

Com o aparecimento dos novos formatos de vídeo de ultra alta definição a par da constante evolução da *Internet*, os muitos benefícios de infraestruturas baseadas em *IP* começaram a despertar o interesse da indústria de televisão. Este interesse é motivado por razões económicas e pelo aumento da flexibilidade e interoperabilidade de toda a cadeia de televisão.

Enquanto que as anteriores infraestruturas baseadas em *SDI* têm que ser reconstruidas cada vez que um novo formato surge, esta nova abordagem baseada em *IP*, pretende ser facilmente adaptável a diferentes formatos de forma a que no máximo apenas seja necessário um *upgrade* do *software* ou mesmo de algum do *hardware* para suportar os diferentes formatos que já existem e os que surgirão no futuro [22].

Neste âmbito, o principal debate da indústria televisiva centra-se nas diferentes abordagens para a uniformização do transporte de vídeo sobre *IP*. Neste momento existem duas abordagens. A primeira envolve o transporte de *media* sobre uma infraestrutura IP após ser processada pela

camada *SDI* tradicional, ou seja uma infraestrutura híbrida *SDI-IP*, enquanto a segunda pretende basear-se unicamente em *IP*.

A primeira abordagem é a que já está a ser mais largamente implementada atualmente e está definida na norma *SMPTE 2022* [23] explicada em 2.6.2. A vantagem, é que o provedor de serviços pode utilizar grande parte do *Hardware* existente na sua infraestrutura baseada em *SDI* tendo apenas que implementar alguns *updgrades* à mesma, tal como adicionar um módulo de encapsulamento e encaminhamento de pacotes *IP* aos módulos de *SDI* já existentes e fazer as adaptações necessárias em *software*.

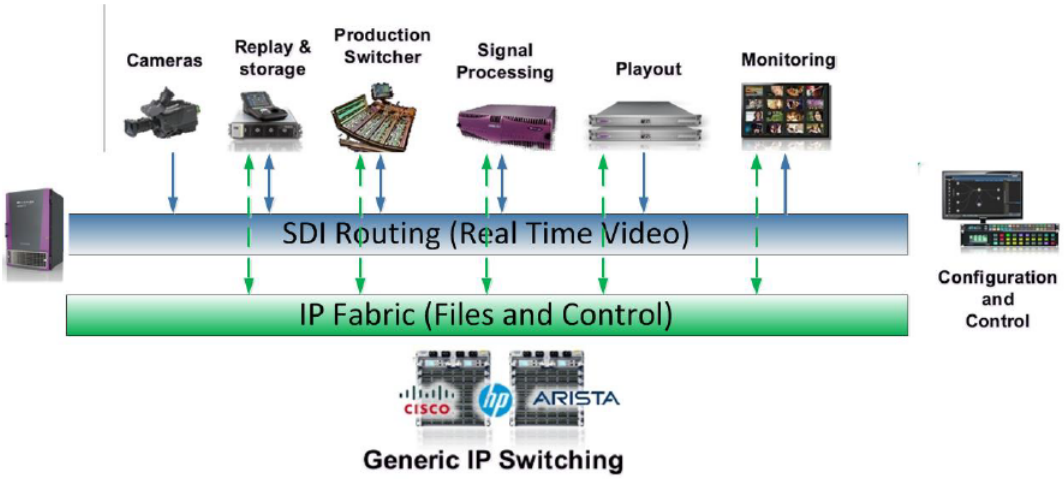

Figura 2.20: Exemplo de uma infraestrutura de produção de televisão baseada na abordagem híbrida *SDI-IP*. [6]

A segunda abordagem pretende eliminar toda a camada *SDI* e utilizar unicamente soluções *IP*. A desvantagem desta abordagem é que são necessários novos componentes tanto na estação de envio, como na estação de receção de sinal, o que implica investimentos elevados e consequentemente mais tempo até que uma grande parte do mercado adote esta solução. Por outro lado, a grande vantagem advém dos os benefícios que traz em termos de flexibilidade do *workflow* e maior facilidade de manipulação dos sinais de media. São exemplos desta abordagem as especificações *VSF SVIP TR-03- Uncompressed Video over IP without SDI Encapsulation*, *SMPTE RDD 34 – Compressed Video Over IP*, *SMPTE RDD 37- Uncompressed Video, áudio and metada over IP* e a série *Networked Media Open Specifications*. [6]

Com esta abordagem pretende-se revolucionar todo o *workflow* da produção de televisão e conseguir que toda a infraestrutura seja baseada em *IP*.

2.6.1 *RTP: Real-time Transport Protocol*

O protocolo *RTP* é um protocolo de transporte de áudio e vídeo sobre redes IP inicialmente descrito na *RFC 1899* [24] em 1996 e atualizado em 2001 na norma *RFC 3550* [7] pelo *IETF.*.

As aplicações que utilizam o *RTP* fazem-no utilizando-o por cima de outro protocolo de transporte, sendo os mais usuais o *Transmission Control Protocol (TCP)* [25] e o *User Datagram*

Protocol (UDP) [26]. Em aplicações de produção de televisão é utilizado maioritariamente o *RTP* sobre *UDP* pois este protocolo embora não ofereça garantias de entrega de pacotes, os atrasos na transmissão são menores. Para além disso, oferece maior flexibilidade uma vez que dá às aplicações a possibilidade de escolha dos seus próprios mecanismos de controlo conforme as suas exigências. [27]

Através da definição de um formato de trama com um cabeçalho, o protocolo *RTP* implementa funcionalidade para o transporte de dados de extremo a extremo para aplicações com exigências de tempo-real. Este cabeçalho, ilustrado na Figura 2.21, contém informação adicional para a implementação de mecanismos capazes de auxiliar o transporte em tempo real e qualidade do transporte.

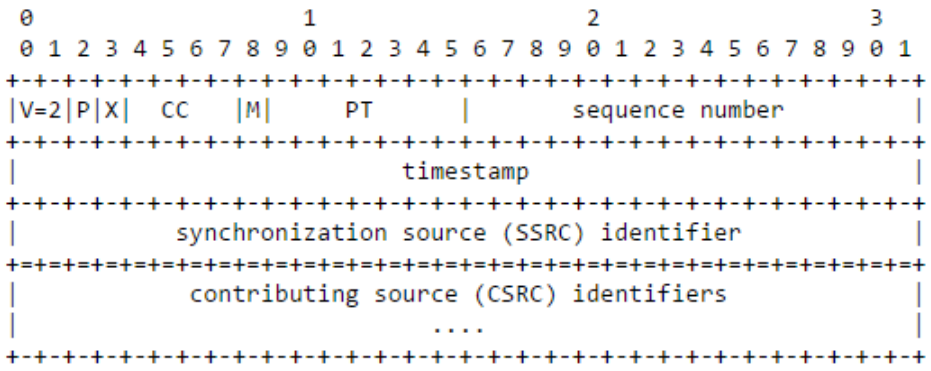

Figura 2.21: Cabeçalho dos pacotes *RTP* segundo a norma *RFC 3550.* [7]

- ***Version (V)*** **-** 2 *bits*: versão do protocolo *RTP*.
- ***Padding (P)*** **-** 1 *bit*: quando este *bit* está a "1" indica que o payload RTP tem no final 1 ou mais octetos de *padding* (enchimento).
- ***Extension (X)*** **-** 1 *bit*: quando este bit está a "1" indica que existe um cabeçalho de extensão.
- ***CSRC count (CC)*** **-** 4 *bits*: indica o nº de identificadores CSRC (*contributing sources*) que seguem o cabeçalho fixo.
- ***Marker (M)*** **-** 1 *bit*: a interpretação deste *bit* é definida em perfis fora do âmbito do *RTP*, pretendendo assinalar a ocorrência de um evento significativo no pacote RTP, como por exemplo o final de um *frame.*
- ***Payload Type (PT)*** **-** 7 *bits*: indica o tipo de dados presentes no *payload.*
- ***Sequence Number*** **-** 16 *bits*: este valor é incrementado de uma unidade por cada pacote RTP enviado. Normalmente é utilizado no recetor para detetar pacotes fora de sequência ou perda de pacotes.

- ***Timestamp-*** 32 *bits*: indica o instante de amostragem do primeiro octeto do pacote *RTP*. Permite sincronização entre emissor ereceptor e cálculo de *jitter*.[3]

- ***Synchronization source (SSRC)-*** 32 *bits*: é um identificador único escolhido de forma aleatória. Duas fontes de sincronização dentro da mesma sessão RTP não podem ter o mesmo identificador.

- ***CSRC identifiers-*** 32 *bits* cada: identificadores das fontes que contribuem para o *payload* do pacote *RTP*. Este campo é opcional, podendo existir no máximo 15 identificadores.

2.6.2 Norma *SMPTE 2022*

Para que fosse possível o uso de tecnologias de comutação de pacotes *IP* em instalações profissionais de *media*, a *SMPTE* desenvolveu a série de *standards 2022*. Iniciado em 2007 e ainda em desenvolvimento à data de escrita desta dissertação, o *SMPTE 2022* focava-se originalmente em vídeo comprimido transportado em *Transport Streams* de *MPEG-2* e depois foi estendido até ser capaz de lidar com vídeo não comprimido, áudio e metadados com débitos maiores que 3 *Gbits/s*.

Hoje, vários fabricantes de equipamento da indústria da televisão implementam este *standard* numa grande variedade de produtos por várias razões: [28]

- É um *standard* global que desde cedo foi abraçado pela índustria;

- Suporta vários formatos de vídeo suportados pelos diferentes sistemas de televisão, incluindo *MPEG-2 TS* e *SDI.*

- É desenhado para todos os tipos de media e tenta corrigir erros que aconteçam durante a transmissão. Uma vez que os erros em redes *IP* normalmente causam a perda de pacotes inteiros e não de apenas um *bit*, os esquemas de correção de erros já utilizados em soluções satélite não iriam resultar. Este protocolo implementa controlo de erros através de um esquema de controlo linha/coluna que é mais adequado para o tipo de erros que ocorrem nestas redes.

Esta família foca-se em tecnologias relacionadas com o transporte de vídeo sobre *IP* e estão pensadas para aplicações de contribuição e distribuição [28]. Cada parte individual desta família define uma funcionalidade específica que é utilizada para criar os pacotes *IP* e também para adaptar esses pacotes aos requisitos técnicos do equipamento de envio e receção dos mesmos.

Desta série destaca-se no âmbito desta dissertação a sexta parte desta norma (*SMPTE ST 2022-6*)[23] que atualmente já é adotada por vários provedores de serviço e que define o transporte sinais de *media* encapsulados em *SDI* sobre *IP*. Esta é baseada no *Real-time Transport Protocol (RTP)* [29] e acrescenta o *High-Bitrate Media Transport protocol* para alta precisão do sinal relógio e inserção de metadados extra para transporte de sinais com um débito maior que *3Gb/s*, como o

[3] Variação estatística do atraso na entrega de pacotes numa rede.

caso de vídeo nos formatos de *UHD*. Segundo esta norma, todo o sinal *SDI* é encapsulado no *payload* dos pacotes de *RTP*.

As diferentes partes do *standard SMPTE 2022* são apresentadas a seguir:

- *ST 2022-1:2007 "Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks"*

Define um mecanismo de *Forward Error Correction (FEC)*[4] através de linhas/colunas para *streams* de vídeo *IP*. Assim como a parte 2, esta tem sido amplamente implementado pela indústria. *FEC* através de linhas/colunas é implementado agrupando pacotes de vídeo *IP* em linhas e colunas lógicas e juntando um pacote *FEC* de verificação a cada linha e a cada coluna. Quando um pacote é perdido referente a uma linha ou a uma coluna, os dados desse pacote podem ser recuperados através da junção dos pacotes *FEC* com os outros dados de cada linha e coluna.

- *ST 2022-2:2007 "Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks"*

Específica como sinais de video de débito constante que são encapsulados em *MPEG-2 TS (transport streams)* são encapsulados em pacotes *IP*. Este *standard* abrange a camada de transporte (*RTP* e *UDP*) assim como sugestões à cerca de temporização e tamanhos do *buffer.*

- *ST 2022-3:2010 "Unidirectional Transport of Variable Bit Rate MPEG-2 Transport Streams on IP Networks"*

Define o empacotamento *IP* para pacotes *MPEG-2 TS* de débito variável que são restringidos a ter débito constante entre mensagens *PCR (Piecewise Constant Rank).*

- *ST 2022-4:2011 "Unidirectional Transport of Non-Piecewise Constant Variable Bit Rate MPEG-2 Streams on IP Networks"*

È similar à parte 3 da norma , excetuando que remove o débito constante de mensagens *PCR*.

- *ST 2022-5:2012 "Forward Error Correction for High Bit Rate Media Transport Over IP Networks"*

È a expansão da parte 1 pois descreve a implementação de *FEC* através de linhas/colunas em vídeos com débito maior ou igual a *3Gb/s*.

- *ST 2022-6:2012 "Transport of High Bit Rate Media Signals over IP Networks (HBRMT)"*

Descreve um protocolo para transporte de conteúdo audiovisual, encapsulado em *SDI*, em tempo real através de redes *IP* com a possibilidade de incluir o mecanismo de *Forward Error Correction (FEC)* descrito na parte anterior. Para além disso inclui ainda *HBRMT (High Bit Rate*

[4] Técnica para controlo de erros em transmissões sujeitas a ruído.

Media Transport) que define o transporte de vídeo sobre *IP* para vídeo de grande exigência de processamento (a partir de 3*Gbps*) , como é o caso do vídeo nos formatos *UHD*. No âmbito desta dissertação, esta é a parte mais importante da norma.

O modelo de camadas para o empacotamento de dados definido por esta parte da norma é apresentado a seguir:

Layer	Abbreviation	Full name	Standard	Length
Application	SDI (payload)	Serial Digital Interface	SMPTE 259M, 292M, 424M	1376
	HBRMT	High Bitrate Media Transport	SMPTE 2022-6	8-16 [a]
	RTP	Real-Time Transport Protocol	RFC 3550	12 [a]
Transport	UDP	User Datagram Protocol	RFC 768	8
Internet	IP	Internet Protocol (v4/v6)	RFC 791 / RFC 2460	20/40 [a]
Link	MAC	Media Access Control (e.g. Ethernet)	IEEE 802.3	42

a. Plus optional fields

Figura 2.22: Modelo de camadas do *SMPTE 2022-6* [8]

- *ST 2022-7 "Seamless Protection Switching of SMPTE ST 2022 IP Datagrams"*

Descreve uma forma de enviar duas *streams* de pacotes redundantes de uma fonte a um destino sobre diferentes caminhos e com sincronização automática entre eles conforme a necessidade. Assim, é possível que o sinal seja reconstruído sem qualquer perda no recetor a não ser que os dois caminhos falhem em simultâneo.

2.6.3 *RFC 4175 - "RTP Payload Format for Uncompressed Video"*

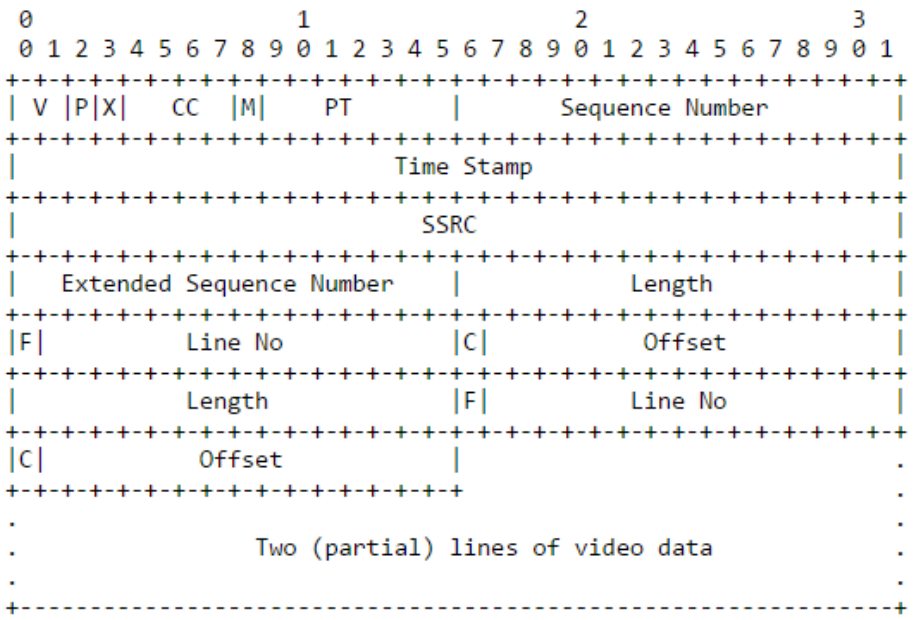

Figura 2.23: Formato de um pacote RTP segundo a norma *RFC4175* com duas linhas de vídeo (parciais). [9]

A norma *RFC 4175*[9] específica um esquema de encapsulamento para vídeo não comprimido para ser enviado sobre *RTP*. Este suporta uma gama de formatos de definição *standard* e de *HD* e foi desenhado para ser extensível a novos formatos aquando do seu aparecimento. Embora tenha sido escrito em 2005, este documento foi desenhado para suportar vídeo com até 32767 linhas por 32767 colunas, o que excede largamente até o maior formato de ultra alta definição, o *UHDTV2* (7680 linhas por 4320 colunas).

O cabeçalho do *payload* especificado no *RFC4175* contém uma extensão de 16 *bits*, *"Extended Sequence Number"*, ao *"Sequence Number"* do cabeçalho *RTP*[7], também de 16 *bits*. Estes dois campos combinados fazem com que seja possível valores muito mais altos do número de sequência dos pacotes, permitindo enviar ou receber pacotes de conteúdos que exigem grande débito, identificando-os sem ambiguidade. Por exemplo, para um vídeo com débito de 1.1 *Gbit/s*, utilizando pacotes de 1500 octetos, é necessário enviar perto de 100 000 pacotes por segundo. O *sequence number* de 16 bits dos pacotes *RTP standard*, voltaria a 0 em cerca de 0,7 segundos. Com a implementação do *RFC4175* isto só aconteceria passado mais de 10 horas.[30]

O cabeçalho especificado, contém ainda o número da linha que está a ser enviada no pacote (campo "Line no", 15 *bits*) , o número de píxeis referentes a essa linha contidos nesse pacote (campo "Length", 16 *bits*) e a posição do primeiro píxel da linha (campo Offset, 15 *bits*). O campo de 1 bit *"Field Identification" ("F")*, é posto a 1 caso o varrimento do vídeo seja interlaçado ou 0 caso seja progressivo.

Como um pacote *RTP* pode conter píxeis de mais do que uma linha, existe ainda o campo de 1 bit "Continuation" ("C"), cujo valor deve ser 1 caso esta condição se verifique. Neste caso, os campos do Payload Header: "Length", "Line no." ,"Offset", "Field Identification" e "Continuation" são repetidos, devidamente adaptados à nova linha.

Os campos dos cabeçalhos *RTP* definidos pela norma *RFC 3550* têm o seu significado habitual, com as seguintes notas para os seguintes campos:

- ***Payload Type (PT):*** 7 *bits* - Deve ser do tipo dinâmico, isto é, entre 96 e 127 segundo o *RFC 3551 - "RTP Profile for Audio and Video Conference with Minimal Control"* [31].

- ***Timestamp:*** 32 *bits* - Indica o instante em que o frame foi amostrado no caso do varrimento do vídeo ser progressivo, ou instante de amostragem de cada *field* no caso do varrimento do vídeo ser interlaçado. No primeiro caso todos os pacotes do frame têm o mesmo *timestamp* e no segundo todos os pacotes do mesmo *field* têm o mesmo timestamp. Em ambos os casos deve-se utilizar um timestamp de 90-*kHz*.

- ***Marker (M):*** 1 *bit* - No caso do vídeo ter varrimento progressivo, este campo no último pacote do frame deve ser 1 e 0 nos restantes. No caso do vídeo ter varrimento interlaçado, este campo no último pacote de cada field deve ser 1 e 0 nos restantes.

- ***Sequence Number:*** 16 *bits* - Corresponde aos bits de menor ordem da versão extendida específicada neste documento anteriormente referida. Complementado este valor com o *Extended Sequence Number* consegue-se um número de sequência de 32 *bits*.

Na Figura 2.23, é apresentado um exemplo para o caso em que duas linhas parciais são encapsuladas num só pacote. È relevante notar-se que os campos *"Length","F","Line no" ,"C" e "Offset"* aparecem em duplicado pois existem duas linhas. Neste caso o valor do primeiro *"C"* tem de ser 1.

Um conceito chave introduzido nesta norma é o conceito de *pixel group* (*(pgroup)*. Os *pgroups* servem para assegurar que num pacote *RTP* está contido um número inteiro de píxeis e um número inteiro de *bytes* sem que seja necessário incluir *bits* de enchimento. Assim, os píxeis que partilham valores devem ser transportados em conjunto e o número de *bits* desse conjunto tem que ser divisível por oito. Por exemplo, num vídeo com subamostragem de cor *4:2:2* e 10 *bits* de profundidade, os *pgroups* são constituidos por dois píxeis (5 octetos).

2.6.4 *JT-NM Reference Architecture v1.0*

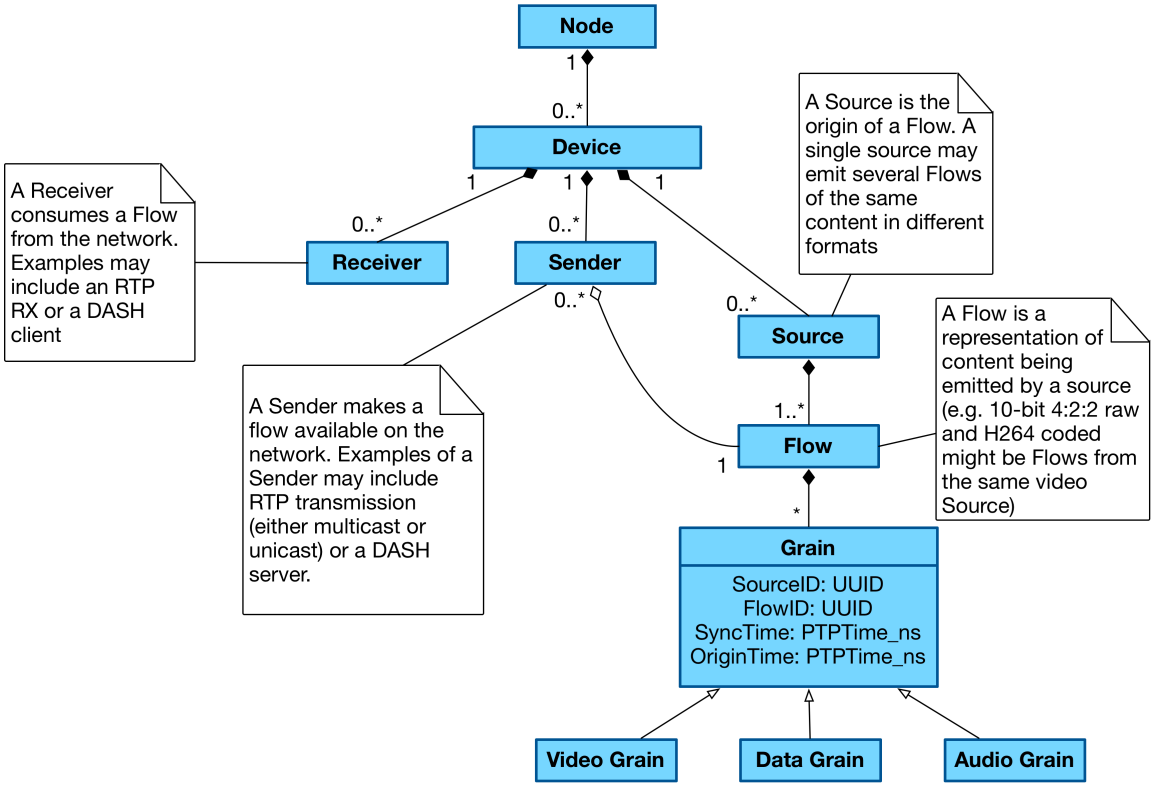

Figura 2.24: Ilustração simplificada do modelo conceptual simplificado introduzido no *JT-NM Reference Architecture v1.0* [10]

Esta arquitetura de referência foi definida em Setembro de 2015 pela *Joint Task Force on Networked Media (JT-NM)* que nasce de uma parceria entre a *European Broadcasting Union*, a *SMPTE* e a *Video Services Forum* (*VSF*). O objetivo desta parceria prende-se com ajudar a adaptação das infraestruturas atuais de produção de televisão para infraestruturas baseadas em *IP* e para isso desenvolvem modelos de referência, *frameworks* e definem boas práticas para ajudar à interoperabilidade dos novos equipamentos dos diferentes fabricantes. No caso deste documento em específico, o mesmo define um modelo conceptual que pretende definir como diferentes equipamentos e conteúdos se relacionam, são identificados e se mantém sincronizados num *workflow* de produção baseado em *IP*.

A ideia pressuposta de "diferentes conetores para diferentes sinais"presente nas infraestruturas tradicionais, é substituída pelo uso de *interfaces* lógicas em *interfaces* de redes comuns, tornando a comunicação mais simples e uma maior flexibilidade da manipulação dos diferentes *media*. Os módulos do *JT-NM Reference Architecture* são descritos a seguir e o seu modelo concetual é apresentado na Figura 2.24 de forma resumida tendo em conta o contexto desta dissertação.

- ***Nodes***

Os *nodes* são *hosts* lógicos para processamento e operações de rede, funcionando como o "cérebro"do modelo. Estes podem já existir fisicamente, ou podem ser criados virtualmente, como por exemplo uma máquina virtual num *cluster* ou nuvem.

- ***Devices***

Os *Nodes* disponibizam *devices*. Estes podem também ser virtuais ou físicos e possuem capacidade de processamento de funções específicas, como por exemplo uma câmara, uma placa de captura de vídeo, ou um módulo de *software* de encapsulamento de sinal *SDI* em pacotes *IP*.

- ***Sources***, ***Flows*** e ***Grains***

Source é um conceito abstrato introduzido para identificar a origem de conteúdos provenientes de um *device* capaz de os gerar. Um *device* pode ter várias *sources*, como por exemplo no caso de um *dispositivo* de *playback* de vídeo que disponibiliza vários *outputs* com vídeos distintos.

Um *flow* é uma sequência de dados provenientes de uma só *source*. Uma *source* pode ter vários *flows*, como por exemplo no caso em que uma câmara é capaz de disponibilizar vários formatos do mesmo vídeo no seu *output*.

Finalmente, os *grain* representam as partes elementares de um *flow*, tal como um *frame* no caso do *flow* representar um vídeo. Um *grain* deve conter meta-dados associados tal como a *source* e o *flow* à qual está associado, bem como informação temporal da sua criação e última modificação. Estas marcas temporais, ou *timestamps*, devem ser derivados do *clock* do *node* ao qual o *grain* está associado que por sua vez deve estar sincronizado com os *clocks* dos restantes *nodes* da rede.

- ***Senders*** e ***Receivers***

Para os conteúdos serem transportados na rede tem que existir uma conexão entre pelo menos um *device* que os envia, o *Sender*, e outro que os recebe, o *Receiver*.

2.6.5 *NMOS: Networked Media Open Specifications*

Network Media Open Specifications, *NMOS*, é uma série de especificações desenvolvidas pela *Advanced Media Workflow Association (AMWA)* que pretendem aumentar a interoperabilidade das soluções de televisão baseadas em *IP*. Embora o objetivo da *AMWA* seja o mesmo que o da *Joint Task Force on Networked Media*, este último não especifica modelos concretos para a implementação. Em vez disso, focam-se em desenvolver modelos conceptuais tal como o *JT-NM Reference Architecture v1.0* para iniciativas como a *NMOS*, que faz uso desse modelo, se focar em detalhes práticos para que os modelos idealizados sejam concretizáveis da melhor forma possível.

As especificações desta série estão ainda a ser definidas e para elas pode contribuir qualquer empresa relacionada com a indústria da televisão. Estas procuram maximizar a interoperabilidade entre os diferentes equipamentos de produção de televisão baseados em *IP* dos diferentes fabricantes desde cedo através de uma abordagem aberta e interativa com os participantes. Para cada

fase, para além de ser discutido o *design* e detalhes de cada especificação, pretende-se também testar presencialmente os *frameworks* desenvolvidos junto dos diferentes parceiros.

As especificações em desenvolvimento à data de escrita deste documento são:

- ***Discovery and Registration Proposed Specification***: específica uma *API* para os *nodes* da infraestrutura registarem, identificarem e gerirem os seus elementos.
- ***In-stream Identity and Timing WIP Specification***: especifíca como os conteúdos devem ser transportados em pacotes *RTP* sobre a rede.

2.6.6 *In-stream Signaling of Identity and Timing information for RTP stream Specification*

Esta especificação da *NMOS* é a mais relevante para esta dissertação e por isso nesta secção será explicada em detalhe. A mesma descreve o mapeamento dentro de pacotes *RTP* [7] de identificadores da *Source* e *Flow* relativos ao conteúdo de vídeo ou áudio a transportar, assim como de marcas temporais dos seus *Grains*. Estes dados são incluidos nos pacotes utilizando a extenção do cabeçalho (*Header Extension*) dos pacotes *RTP*.

A especificação é agnóstica ao tipo de conteúdo audiovisual que se insere nos pacotes uma vez que antes da transmissão do vídeo é enviado ao recetor um ficheiro de texto com um formato descrito no *Session Description Protocol (SDP)* preenchido no emissor. O *SDP* está definido na norma *RFC4566* [32] e consiste num protocolo baseado em texto onde se pode definir e descrever uma sessão de transmissão, o tipo de conteúdo a enviar e incluir informação temporal referente à sessão. Os campos do *SDP* são uma sequência de linhas de texto com o formato *«atributo*> = <*valor*> *"*. Na subsecção 3.4.2.2 é apresentado um exemplo de um ficheiro neste formato segundo esta especificação.

À data de escrita desta dissertação, esta especificação ainda não tinha sido concluída e por isso outros protocolos de transporte para além do *RTP* podem ser suportados em revisões futuras.

2.6.6.1 Preenchimento do *RTP Header Extension* segundo a *NMOS*

Para preencher o cabeçalho dos pacotes *RTP* com todos os campos especificados na *NMOS* segue-se o *RFC 5284- "A General Mechanism for RTP Header Extensions"* [33]. Este *RFC* define um mecanismo para que seja possível incluir no *RTP Header Extension* várias extensões mais pequenas.

Segundo a *NMOS* as seguintes extensões são incluídas no *RTP Header Extension*:

1. *PTP Sync Timestamp*
2. *PTP Origin Timestamp*
3. *SMPTE ST 12-1 (SMPTE 12M) Timecodes* (opcional)
4. *Flow Identifier*

5. *Source Identifier*
6. *Grain Duration* (opcional)
7. *Grain Flags*

No primeiro pacote de cada *grain* incluem-se as extensões 1-7 e no última pacote de cada *grain* inclui-se a extensão 7, sendo que nos restantes pacotes não se inclui uma extensão do cabeçalho *RTP*.

O formato do *RTP Header Extension* é mostrado na figura 2.25.

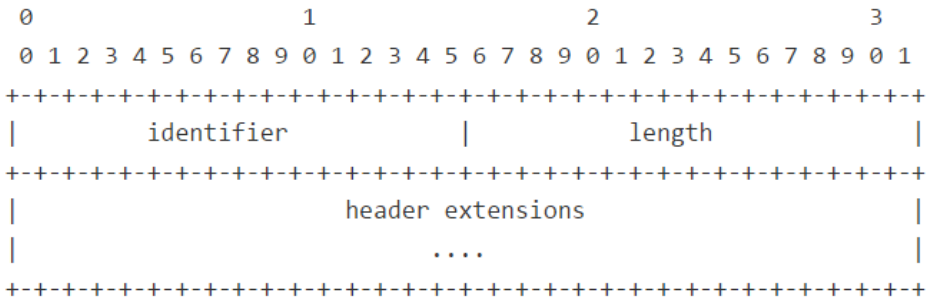

Figura 2.25: Formato do *RTP Header Extension*

O valor do campo *identifier* deve ser "0xBEBE"e o valor do *length* deverá ser igual ao tamanho total das *header extensions* incluidas no pacote. Assim, no caso da *NMOS* no final do cabeçalho extra *RTP* do primeiro pacote de cada *grain*, que tem 76 *bytes* no caso de serem utilizados os campos opcionais, devem ser incluídos 4 *bytes* de enchimento e no final do cabeçalho do último pacote de cada *frame*, que tem 1 *byte*, devem ser incluídos 3 *bytes* de enchimento.

Cada extensão da *NMOS* começa com um cabeçalho de 1 *byte*, conforme representado na Figura 2.26.

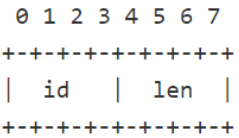

Figura 2.26: Cabeçalho de 1 *byte* específicado na norma *RFC 5285*.

O identificador *ID* é o identificador local de cada extensão e pode assumir valores de 1 a 14. O valor do tamanho *length* é o número de *bytes* de cada extensão menos 1 unidade. Assim, quando este assume o valor 0 indica-se que a extensão tem 1 byte.

O mapeamento entre cada identificador local e o respetivo conteúdo é indicado no ficheiro de texto *SDP* enviado antes da transmissão *RTP*.

A descrição dos campos é a seguinte:

- **Extensões 1 e 2:** *PTP Sync Timestamp Header Extension* e *PTP Sync Timestamp header Extension*

  Cada um destes *timestamp* contém 2 campos: um de 48 *bits* referente aos segundos em que o *frame* foi amostrado , e outro de 32 bits referente aos nanosegundos.

- **Extensão 3** *SMPTE ST 12-1 (SMPTE 12M) Timecode Header extension*

  Tal como o nome indica, esta extensão é preenchida com um *timecode* conforme descrito no *SMPTE ST 12-1* [34] . O valor do mesmo ocupa 8 *bytes* e por isso o seu *length* é de 7. O seu uso é opcional.

- **Extensões 4 e 5:** *Flow Identifier header Extension* e *Gran duration Header Extension*

  Estes cabeçalhos representam um *UUID* conforme especificado na norma *RFC 422: "A Universally Unique IDentifier"* [35]. O *UUID* representa um identificador único com um formato especifico definido na norma e que ocupa 16 *bytes*, logo valor do *length* será 15.

- **Extensão 6:** *Grain Duration Header Extension*

  Este cabeçalho opcional consiste em dois campos que indicam a duração de um *grain*: um numerador e um denominador, cada um de 4 *bytes*. Por exemplo, um vídeo com *frame rate* de 25 teria como valor do numerador 1 e denominador 25.

- **Extensão 7:** *Grain Flag Header Extension*

  Grain Flag é uma extensão de um único *byte* com 3 campos. O primeiro campo *"Start Flag"* de um só *bit* é posto a 1 no primeiro pacote de cada *Grain* e a 0 nos restantes. O *bit* seguinte corresponde ao campo *"End Flag"* e é posto a 1 no último pacote de cada *Grain* e a 0 nos restantes. Os 6 ultimos *bits* são referentes ao campo *"Reserved"* e à data de escrita desta dissertação estavam reservados para uso futuro.

Capítulo 3

Solução Desenvolvida

3.1 Descrição do Problema

Com o aparecimento de cada vez mais câmaras e conteúdos produzidos de Ultra Alta Definição é necessário que os produtores possuam soluções à altura capazes de lidar com os novos formatos. Os formatos de Ultra Alta definição, por permitirem uma qualidade muito superior às dos formatos antecessor, os formatos de *HD*, requerem novas exigências técnicas, nomeadamente a nível de capacidade de processamento pelos equipamentos que o suportam. Para além disso, por serem formatos novos a serem utilizados ainda num caráter exploratório, não existem muitas soluções de produção de televisão no mercado para os mesmos nem estão uniformizadas pela indústria em muitas das suas aplicações.

Um dos desafios introduzidos pelos formatos de *UHD*, surge do tamanho do *payload* do vídeo. Este, durante a fase da produção de televisão, fase em que se enquadra esta dissertação, não sofre compressão para garantir a qualidade máxima possível e para garantir a interoperabilidade com o resto da cadeia de produção em que se insere. O *payload* do vídeo do formato mais pequeno de ultra alta definição, o *UHDTV1*, é até cerca de oito vezes maior do que o *payload* do vídeo do maior formato de vídeo *HD*, o *Full HD* (1920x1080 píxeis): o vídeo *UHDTV1* possui o dobro dos píxeis horizontais, o dobro dos pixeis verticais e só suporta varrimento progressivo. Este desafio torna-se ainda maior no caso dos eventos em direto, em que todo o processamento deve ser feito idealmente em tempo real e sem erros ou perdas de *frames*.

Além do mais, a tendência da indústria da televisão em adaptar as suas soluções de transporte e processamento de vídeo em soluções baseadas em *IP*, embora promissora, é recente e por isso os *standards* referentes às mesmas ainda não estão completamente desenvolvidos. È por isso urgente e extremamente necessário garantir a interoperabilidade entre os diferentes produtos de diferentes fabricantes de modo a que a mesma seja conseguida o mais rapidamente possível.

3.2 Arquitetura da solução

Conforme ilustrado na Figura 3.1, este trabalho foi dividido em duas fases: o desenvolvimento de um módulo de *software* de captura de vídeo de *UHD* por *SDI (“Phase 1")* e o desenvolvimento de um módulo de *software* para envio sobre *IP* do vídeo capturado (*Phase2*).

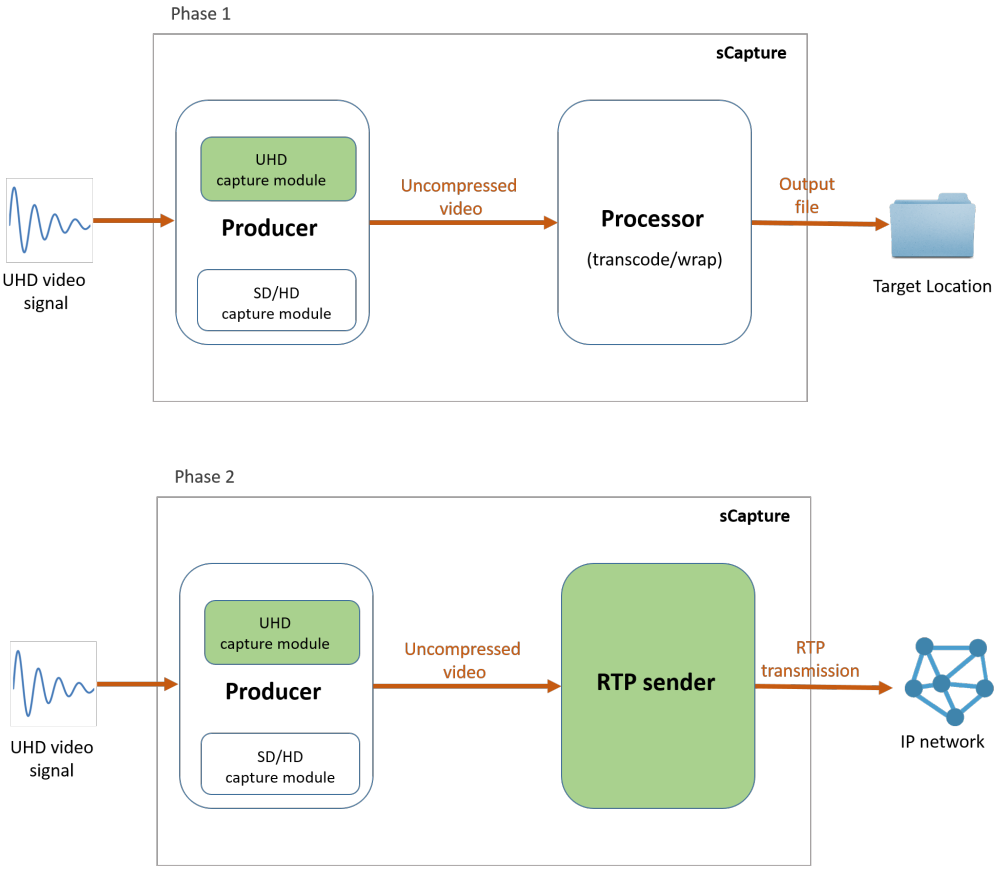

Figura 3.1: Diagrama geral do trabalho desenvolvido

Os módulos de *software* desenvolvidos pela empresa apresentam fundo branco e os módulos a desenvolver nesta dissertação fundo verde: *“UHD capture module”* e *“RTP Sender”*. Assim, a *MOG Technologies* já desenvolveu um módulo de captura de vídeo *SD* e *HD* para este equipamento assim como o módulo *“Processor”* onde acontecem as operações de *transcode/wrapping* para o vídeo ser armazenado localmente. Esse módulo de captura deverá funcionar como referência para desenvolver o de vídeo *UHD* referente à primeira fase desta dissertação.

A gama de produtos da *MOG Technologies* onde os módulos desenvolvidos nas duas fases da dissertação foram incluídos foi a do *mxfSPEEDRAIL* [36]. Esta gama de produtos permite várias funcionalidades do âmbito da produção de televisão nomeadamente captura, *ingest* e *transcoding*. Aquando da realização desta dissertaçaõ, encontrava-se em fase de desenvolvimento uma solução desta gama de produtos responsável para captura de vídeo, à qual nesta dissertação por simplicidade se chamará *sCapture* (explicada em 3.3) e foi onde todos os módulos desenvolvidos foram integrados. Todos os módulos foram escritos na linguagem de programação *C++*.

O *hardware* utilizado para a realização desta dissertação foi um servidor físico onde se instalou o sistema operativo *Windows Server 2012 R2* [37] e uma placa externa de captura/*playback* de vídeo por *SDI DeckLink 4k Pro* [11] da *Blackmagic Design* [38] (os detalhes da placa são apresentados em 3.3.3). Para enviar o vídeo sobre *IP*, na segunda fase desta dissertação, a este servidor foi adicionado uma placa de rede *10 GbE*.

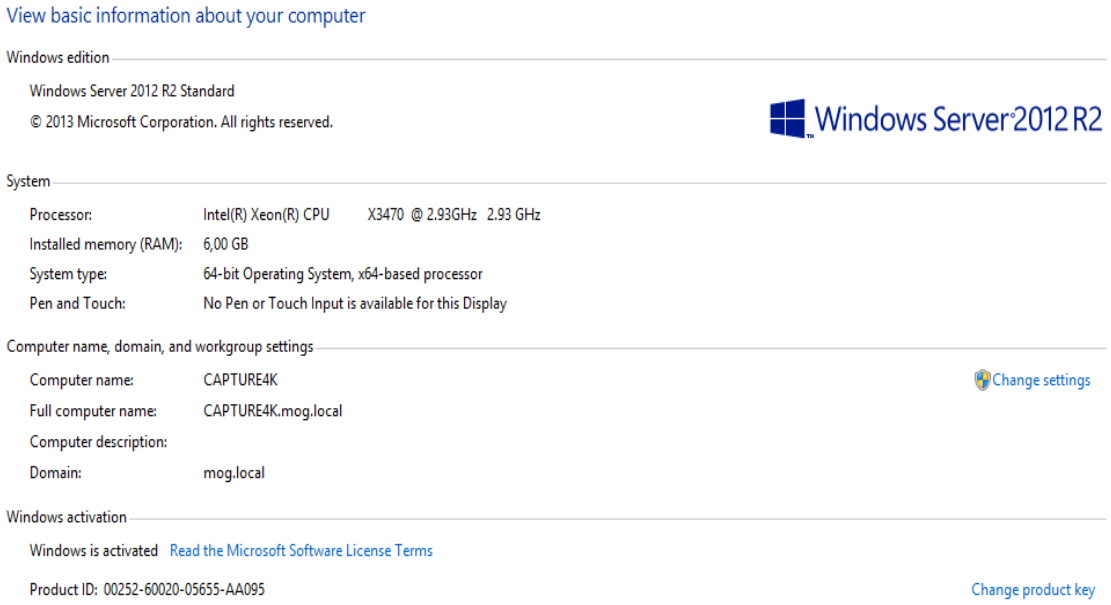

Figura 3.2: Detalhes técnicos do servidor utilizado para desenvolver os módulos desta dissertação.

3.3 Captura de vídeo de *UHD*

Como anteriormente referido, a *MOG Technologies* já possui uma solução para captura de vídeo, nomeadamente a parte de captura de vídeo de formatos antecessores aos de *UHD*. Esta solução, o *sCapture*, integrada no *mxfSPEEDRAIL*, possui uma arquitetura específica com requisitos específicos. Assim, decidiu-se implementar diretamente o módulo de captura de vídeo de *UHD* nesta solução.

O *sCapture* permite:

- Capturar vídeo em *SDI* através de uma placa de captura;
- Gravar o vídeo capturado para um ficheiro *(file-based recording)*;
- Monitorizar o *input* recebido, através de *streaming* de vídeo em *MPEG-DASH;* [1].

[1] Dynamic Adaptative Streaming over HTTP (DASH) ou *MPEG-DASH* é um *standard* para *streaming* de conteúdo audiovisual sobre *IP* em servidores *web* convencionais

- Monitorizar e controlar um ou mais canais *SDI*, permitindo a monitorização de vários vídeos capturados em tempo real e a gravação dos mesmos.

A arquitetura do *sCapture* é apresentada de forma resumida (tendo em conta o âmbito da dissertação) na Fig 3.3.

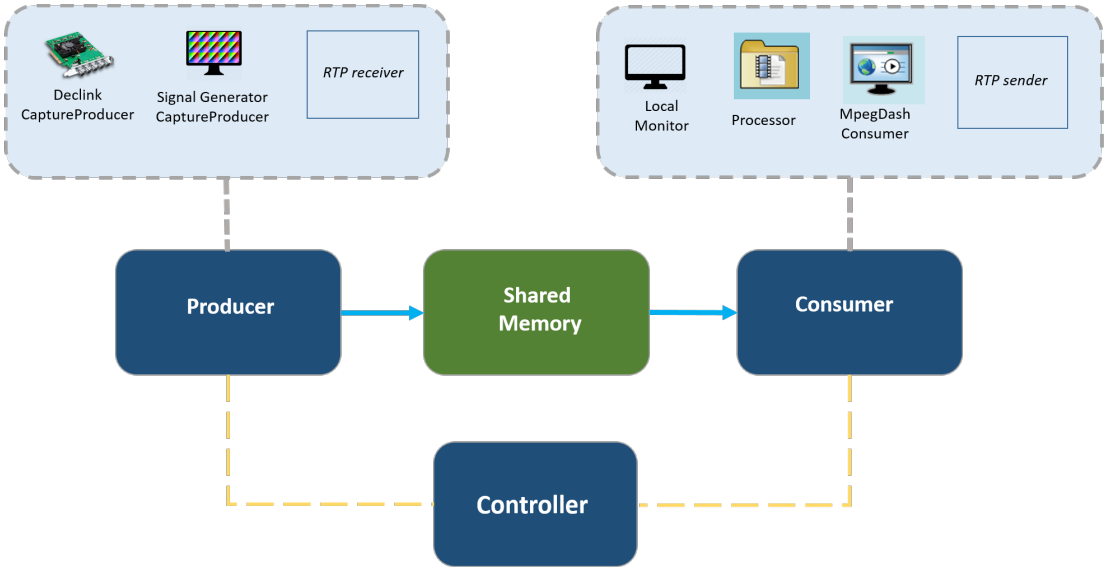

Figura 3.3: Arquitetura da solução de captura inserida no *mxfSPEEDRAIL*: o *sCapture*.

O *sCapture* está desenhado com uma arquitetura multi-processo, o que significa que os grupos de módulos são independentes entre si e operam simultaneamente no tempo. Cada instância do *sCapture* pode ser dividida em três grupos principais:

- ***Producer:*** responsável por capturar o sinal de *input* e gravá-lo na memória partilhada *(Shared Memory)*. O sinal pode ser:
  - recebido em *SDI* através de uma placa de captura *DeckLink 4k Pro* - módulo ***DeckLink Capture Producer.***
  - gerado por um gerador de sinal virtual criado para efeitos de testes- módulo ***Signal Generator.***
  - recebido por *RTP* - módulo ***RTP receiver***. Este módulo ainda não existia no *sCapture* e foi parcialmente implementado no âmbito desta dissertação, para efeitos de testes, como explicado em 3.4.1.2.
- ***Consumer:*** responsável por ler o conteúdo que o *Producer* escreve na *Shared Memory* e processá-lo de acordo a configura ção escolhida pelo utilizador. O ***Consumer*** pode ser um dos diferentes módulos:
  - ***Processor:*** dada a configuração do *workflow*, grava o conteúdo audiovisual capturado para um ficheiro.

  - ***MpegDash Consumer:*** Cria um *proxy* (versão de baixa resolução) do vídeo que está a receber para ser apresentado numa página *web* através de *MPEG-DASH*.
  - ***RTP sender***: Envia o vídeo presente na *Shared Memory* sobre *RTP*. Este módulo foi desenvolvido nesta dissertação.

- **Controller:** serve de ponte entre o *Producer* e o *Consumer*, configurando-os de acordo com as operações escolhidas pelo utilizador que interage diretamente com este módulo para controlar a sessão de captura.

3.3.1 Subfase 1: Implementação *Signal Generator* e *Producer*

O papel do *Producer* é escrever na *Shared Memory* vídeo encapsulado em *SDI* gerado por uma fonte de sinal, como por exemplo uma câmara. No entanto, para esta subfase, foi desenvolvido um gerador de sinal de vídeo *UHD* virtual, o módulo *Signal Generator*, para introduzir à entrada do *workflow*. Deste modo, ao garantir que todo o *workflow* funciona com este input virtual, quando se introduzir uma fonte de sinal *SDI* à entrada do *workflow* pode-se garantir que tudo o que está a jusante deste módulo irá funcionar corretamente.

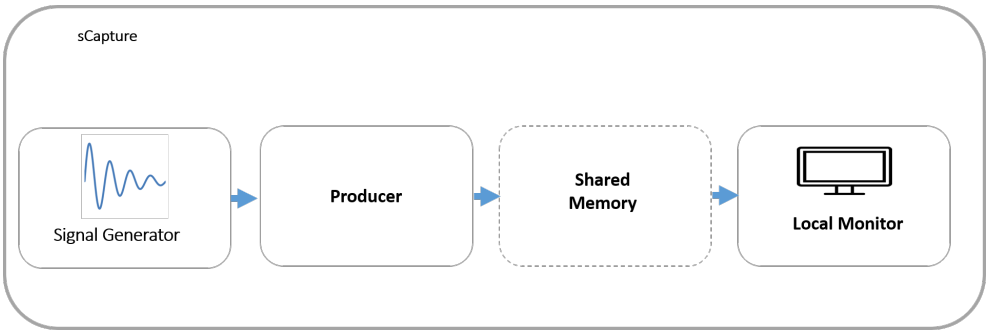

Figura 3.4: Diagrama do *workflow* dos módulos da Fase 1: Implementação *Signal Generator* e Producer

Começou-se por realizar testes unitários com formatos *SD* para compreender o modo de funcionamento do *Signal Generator.*

Ao inicializar o gerador de sinal, um dos parâmetros necessários é um descritor do vídeo que queremos gerar em termos de formato, profundidade de cor, subamostragem de cor, entre outros. No caso específico dos formatos de *UHD* o gerador de sinal foi configurado para gerar todos os formatos de *UHDTV1*, *UHD DCI* (*4096x20196*) e *UHDTV2*.

Depois de inicializar o gerador de sinal com todos os parâmetros necessários, procedeu-se à inicialização e configuração do *Producer* para que escreva o vídeo gerado na *Shared Memory*.

No final, foram realizados testes unitários para todos os novos formatos implementados recorrendo-se ao módulo *Local monitor*. Recorrendo-se a este módulo foi possível avaliar visualmente o sucesso da implementação.

3.3.2 Subfase 2: Integração com o módulo *Processor (wrap/transcode)*

Nesta subfase foi necessário aprender o modo de funcionamento do bloco *Processor* de forma a conseguir adaptá-lo a formatos de *UHD*. Este módulo ainda não suportava estes formatos e essa tarefa não se encontrava nos objetivos da dissertação. Assim, foi feito o *downscaling*, isto é, a redução do tamanho do formato, dos vídeos de *UHD* para o formato de *HD 1080p* através de uma função já utilizada pela empresa para o efeito. Após esta redução e inicializado e configurado o *Processor*, foram realizados testes unitários guardando os vídeos no formato de *UHD* gerados pelo *Signal Generator* em disco e lendo-os de seguida através de um *media player*.

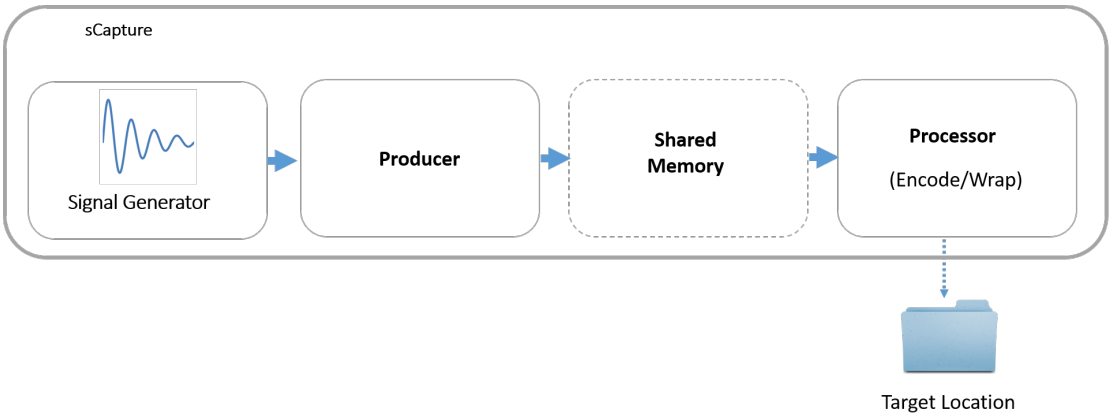

Figura 3.5: Diagrama do *workflow* dos módulos da Fase 2: Integração com o módulo *Processor (wrap/transcode)*

3.3.3 Subfase 3: Integração com a placa de captura *DeckLink 4k Pro*

Esta é a subfase mais relevante da captura uma vez que é implementado na totalidade o caso relativo à primeira parte da dissertação: a captura de vídeo de *UHD* recebido por *SDI*.

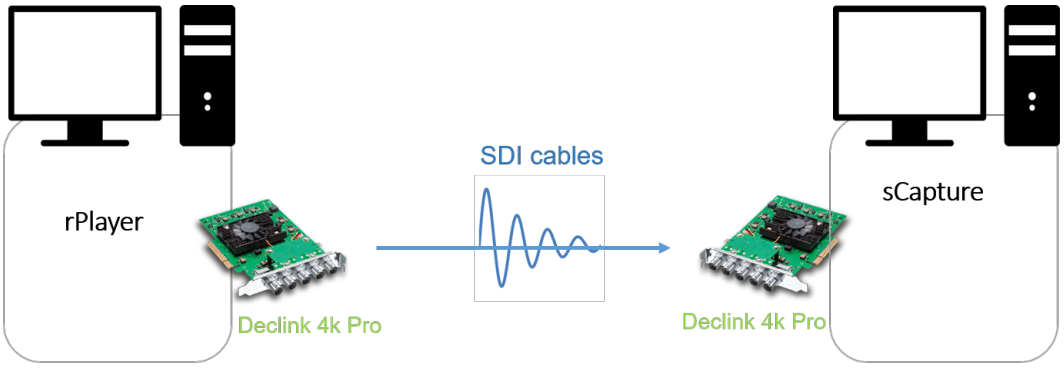

Figura 3.6: Diagrama geral do *workflow* da Fase 3

Em ambos os servidores que intervêm nesta subfase foi incluída uma placa *DeckLink 4k Pro* [11] que serve de *interface* entre o sevidor e o cabo *SDI*.

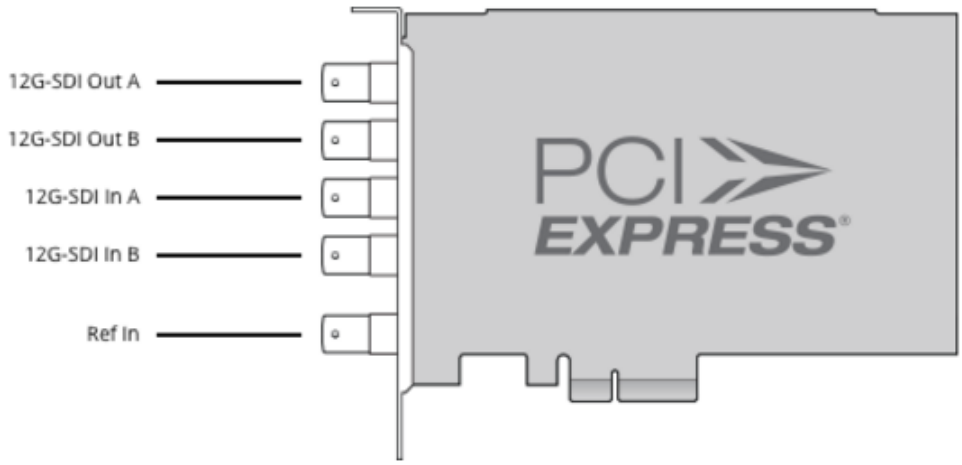

Figura 3.7: Esquema de entradas e saídas da *DeckLink 4k Pro* [11]

Esta placa possuí duas entradas e duas saídas *12G-SDI*, podendo ser utilizadas em *single link* (um só cabo transporta todo o conteúdo audiovisual) ou *dual link* (dois cabos *SDI* transportam partes do mesmo conteúdo).

Os formatos suportados pela placa são apresentados na Figura 3.8.

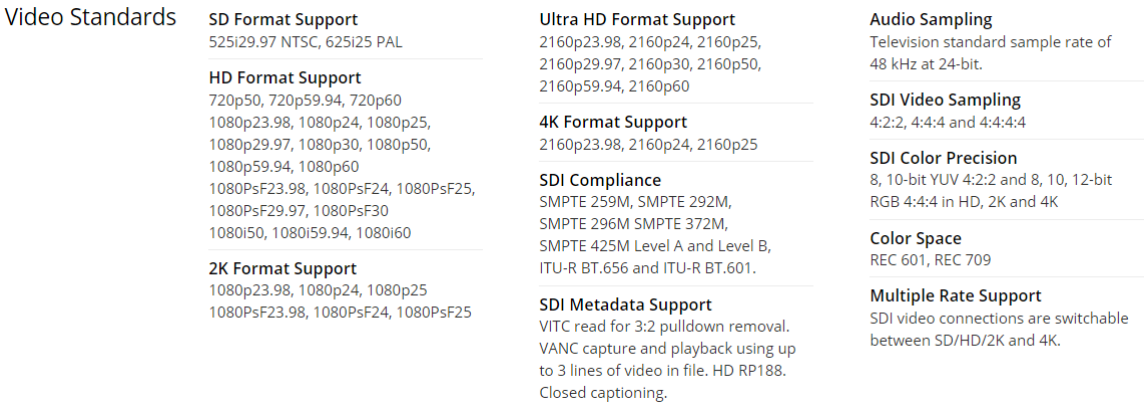

Figura 3.8: Formatos suportados pela *DeckLink 4k Pro* [11]

Para manipular a *DeckLink 4k Pro* foi utilizado o *Decklink SDK 10.4.1* que é um *Software Development Kit* baseado em *C++*, facultado pelo fabricante da placa de captura utilizada que fornece *interfaces* e bibliotecas para controlo de alto e baixo nível de conteúdo audiovisual.

O *input* da captura seria idealmente uma câmara capaz de captar formatos *UHD* e disponibilizar esse conteúdo por um *output SDI*. No entanto, como as câmaras profissionais de *UHD* têm um custo elevado, foi utilizada uma solução de *software* de *playout* de vídeo sobre *SDI* em desenvolvimento pela *MOG-Technologies* que nesta dissertação por simplicidade se chamará *rPlayer*.

O *rPlayer* é então uma solução de *software*, escrita em *C++*, que permite ler ficheiros de vídeo alojados localmente e transmiti-los por *SDI* utilizando uma *interface* tal como a *DeckLink 4k Pro*.

Assim como o *sCapture*, este *software* foi instalado num servidor físico. O sistema operativo deste servidor é também o *Windows Server 2012 R2* e as suas especificações técnicas são as mesmas que as do servidor onde foi desenvolvido o *sCapture* (ver Fig 3.2) à excepção da memória *RAM* que é de 8 *GB* em vez dos 6 *GB* desse servidor.

Deste modo, a primeira etapa desta fase foi adaptar o *rPlayer* para conseguir ler vídeos de *UHD*. Esta adaptação passou por descrever todos os tipos de vídeo *UHD* de acordo com os modelos internos do *rPlayer*. De seguida adaptaram-se os novos modelos implementados para serem enviados pela *DeckLink 4k Pro* recorrendo-se ao *Decklink SDK 10.4.1*.

Depois de adaptar o módulo para todos os formatos suportados pela placa, foram realizados testes unitários através da ligação por cabos *SDI* do servidor onde este módulo foi incluído ao outro onde foi desenvolvido o *sCapture*. Para isso, neste servidor responsável pela captura instalou-se o *software Media Express* [39] da *Blackmagic* que permite visualizar e capturar o vídeo à entrada da *DeckLink 4k Pro*. Visualizando e capturando o vídeo de *UHD* gerado pelo *rPlayer* recorrendo-se ao *Media Express* foi possível testar o *rPlayer*.

No servidor de captura, para que se conseguisse capturar o vídeo pela placa de captura através do *sCapture*, depois de instalada a *DeckLink 4k Pro* e o *DeckLink SDK 10.4.1*, foi necessário manipular o vídeo de modo a adaptar os formatos do *SDK* aos formatos internos do *sCapture*. O restante *workflow* do *sCapture* já tinha sido totalmente desenvolvido e testado nas fases anteriores.

No final, foram realizados testes unitários para todos os formatos implementados enviando vídeo encapsulado em *SDI* através do *rPlayer* e através do *sCapture* capturando-se, visualizando-se e fazendo-se o *ingest* para o disco local em simultâneo. Para isto recorreu-se a um servidor *web* desenvolvido pela *MOG Technologies* só para efeitos de testes dos módulos do *mxfSPEEDRAIL*. Este servidor é acedido através do *browser* e possui uma *interface gráfica* que permite configurar o *controller* para controlar a sessão de captura e que inclui o *MpegDash Consumer* para visualizar o vídeo capturado em tempo real.

Assim, foi concluído todo o *workflow* da primeira fase desta dissertação: a captura de vídeo *UHD* em tempo real.

3.4 Envio de vídeo de *UHD* sobre *RTP*

3.4.1 Desenvolvimento de uma solução simples de transporte de vídeo sobre *RTP*

Contrariamente ao que aconteceu no desenvolvimento do módulo da primeira fase "Captura de vídeo *UHD*" que foi desenvolvido diretamente no *sCapture* do *mxfSPEEDRAIL*, a solução de transporte de vídeo sobre *RTP* foi desenvolvida completamente à parte deste e só posteriormente se procedeu à integração no produto. Tal decisão deveu-se ao facto de na *MOG Technologies* à data do desenvolvimento deste módulo, ainda não existirem soluções de transporte de vídeo sobre *RTP* e por isso haver muito mais decisões a tomar no âmbito. Estas decisões seriam mais fáceis de tomar num projeto à parte desenvolvido do zero que se destinasse única e exclusivamente a este módulo.

Nesta subfase, para processamento e manipulação de vídeo escolheu-se utilizar a biblioteca *OpenCV (Open Source Computer Vision Library)* [40]. Esta biblioteca, escrita em *C++*, possui módulos de processamento de imagens e vídeo, estrutura de dados, álgebra Linear, *GUI* (Interface Gráfica do Usuário) em janelas independentes, controlo do rato e visão computacional. A biblioteca foi desenhada pensando na otimização da eficiência computacional com um grande foco para aplicações de tempo real. Para além de cumprir todos os requisitos técnicos e legais necessários para a realização desta dissertação, este biblioteca conta com uma grande comunidade estimada em mais de 47 mil utilizadores e um número de *downloads* superior a 9 milhões.

Para a transmissão *RTP* utilizou-se a biblioteca *JRTPLIB* [41]. Uma biblioteca de código aberto desenvolvida por *Jori Liesenborgs* da *Hasselt University*, orientada a objetos e escrita em *C++* para suporte de aplicações que utilizam o *RTP* definido no *RFC 3550* por cima de *UDP*. Esta biblioteca disponibiliza módulos de envio e recepção de pacotes *RTP*, manipulação de pacotes, configuração da sessão *RTP* e mecanismos controlo utilizando o protocolo *RTCP (Real-Time Transport Control Protocol)*. [2]

Esta solução está dividida em dois módulos: o módulo principal, o de envio de vídeo sobre *RTP* ("*RTP sender*") e o módulo de receção do mesmo (*"RTP receiver"*) que embora não fosse parte dos objetivos da dissertação, foi desenvolvido para efeitos de testes de forma garantir que o *RTP sender* funcionava corretamente.

3.4.1.1 Módulo de envio: *RTP sender*

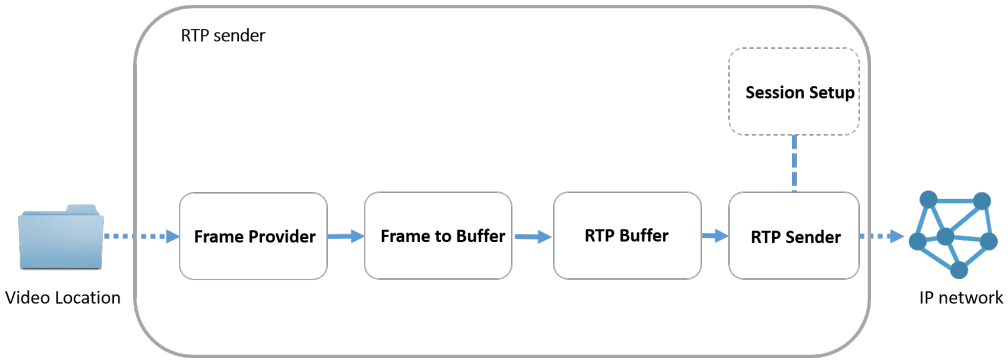

Figura 3.9: Diagrama do *workflow* do módulo de envio de vídeo sobre *RTP ("RTP sender")*

O módulo de envio de vídeo sobre *RTP*, o *RTP sender*, está dividido em submódulos conforme mostrado na Figura 3.9.

O primeiro submódulo é o da leitura de vídeo, ***Frame Provider***. O vídeo, que se encontra em disco, é lido *frame* a *frame* através das funções disponibilizadas pela *OpenCV*. Posteriormente, cada *frame* individual é manipulado no módulo ***Frame to Buffer*** de modo a ser integralmente

[2] Procolo definido na recomendação *RFC 3550* para monitorização da sessão *RTP* através do envio periódico de pacotes de controlo.

guardado num *buffer* de uma só linha. Passando-se de uma matriz de píxeis, para um vetor unidimensional de tamanho igual ao produto do número de linhas com o número de colunas dessa matriz. Este número de linhas e de colunas depende não só do formato do vídeo como também da amostragem e profundidade de cor do mesmo.

Segundo a recomendação *RFC 3550* o tamanho dos dados transportados em cada pacote *RTP* não deve ser superior a 1500 *bytes*. Assim, no módulo ***RTP Buffer*** começa-se por dividir o *buffer* que contém a informação integral do *frame* a transportar em *buffers* mais pequenos para poderem ser transportados sobre *RTP*. Esta divisão é feita de acordo com a recomendação *RFC 4175* e por isso os píxeis são agrupados em *pgroups*.

De seguida, no submódulo *Session Setup* é configurada e inicializada a sessão de transmissão *RTP* recorrendo-se ás funções da biblioteca *jrtplib*. Esta configuração inclui a indicação do *IP* de destino, a porta por onde os pacotes serão enviados, o tamanho do *buffer* de transmissão utilizado pela placa de rede para envio de pacotes e o valor do incremento do (*timestamp* entre cada pacote.

Por fim, no submódulo ***RTP Sender*** começa-se por preencher o *payload* do pacote *RTP* a enviar de acordo com os parâmetros especificados no *RFC 4175*. Junta-se depois ao vetor do *payload* a parte referente ao vídeo contida no *buffer* recebido pelo submódulo *RTP Buffer*. Finalmente preenche-se o *header* do pacote e envia-se sobre *RTP* recorrendo às funções da biblioteca *jrtplib*.

Todo o *workflow*, á exceção do módulo *Session Setup* que só necessita de ser chamado uma vez, é repetido até que o vídeo seja completamente enviado.

3.4.1.2 Módulo de receção: *RTP receiver*

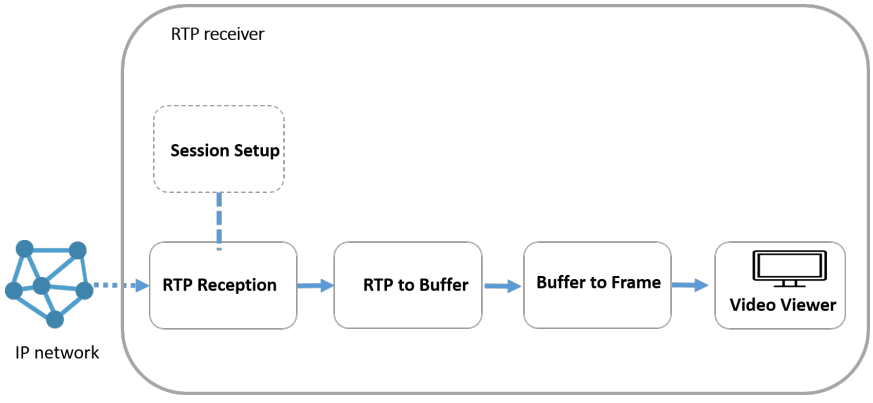

Figura 3.10: Diagrama do *Worklow* dos submódulos pertencentes módulo de receção de vídeo sobre *RTP: RTP receiver*.

Este módulo de receção de vídeo sobre *RTP* é complementar do primeiro e por isso muitas das operações que aqui acontecem são idênticas às desse mas pela ordem inversa.

O primeiro passo é portanto configurar a sessão *RTP* recorrendo-se também ás funções da biblioteca *jrtplib*. Esta configuração agora inclui a indicação da porta por onde os pacotes vão ser recebidos, o tamanho do *buffer* de receção utilizado pela placa de rede para receção de pacotes e a

indicação de que se deseja esperar até 10 minutos pela receção de pacotes. Este é também o único submódulo do *"RTP receiver"*que só é chamado uma vez.

Depois da configuração da sessão já se pode receber os pacotes. Para que isso aconteça utilizaram-se funções do *JRTPLIB* e o submódulo responsável é o ***RTP Reception***. Este submódulo, depois da receção do primeiro pacote, se o mesmo corresponder ao formato esperado, aceita unicamente os pacotes desse *IP*. O formato esperado, nesta fase, seria apenas o campo *"marker"* do cabeçalho do primeiro pacote *RTP* correspondente ao primeiro *frame* ter o valor "1". Como será mostrado à frente, nas outras fases do projeto este primeiro pacote foi sinalizado segundo o especificado na *NMOS*.

Após avaliado o campo *marker* e identificado o *IP* de origem dos pacotes, prossegue-se à extração dos dados do pacote no submódulo ***RTP buffer***. Neste módulo, é preenchido um *buffer* à medida que os pacotes *RTP* são recebidos, quando todos os pacotes referentes a um *frame* são recebidos, o *buffer* fica completo. Procede-se então à transformação do *buffer* de uma só linha para uma matriz de três dimensões referente a um *frame* de *M* linhas e *N* colunas, consoante as características do vídeo.

Por fim, através do submódulo ***Video viewer***, os *frames* são mostrados numa janela à medida que são recebidos, recorrendo-se à biblioteca *OpenCV* para o efeito.

Esta solução tinha como objetivo sobretudo escolher a melhor biblioteca a utilizar para transporte de vídeo, estudar e compreender a mesma e perceber como deveria ser feito o transporte e a manipulação do vídeo. A solução foi testada com ambos os módulo a correr em paralelo na mesma máquina com vários formatos de vídeo, nomeadamente formatos de *UHD*.

3.4.2 Implementação segundo a *NMOS*

Como referido em 2.6, as soluções atuais consideradas como as mais relevantes para a indústria da televisão para o transporte de vídeo não comprimido sobre *IP* são as defenidas na série *SMPTE 2022* e a especificação *NMOS Mapping of Identity and Timing Information to RTP*.

Para esta dissertação, escolheu-se utilizar a segunda por diversas razões. Em primeiro lugar porque segundo o *SMPTE 2022-6* ao se encapsular todo o sinal *SDI* há uma grande quantidade de dados auxiliares referentes ao mesmo que podem ser eliminados, podendo este valor chegar a 40% dependendo do formato e do *frame rate* do vídeo [42]. Além do mais, a flexibilidade e fiabilidade das infraestruturas de produção pode ser aumentada separando o vídeo, o áudio e dados auxiliares para serem transportados em fluxos de dados individuais evitando que seja necessário desmultiplexar e multiplexar todo o sinal *SDI* sempre que seja preciso um dos elementos. Esta separação está definida na especificação da *NMOS*. A utilização do *SMPTE 2022-6* pode fazer sentido em muitas infraestruturas de televisão atuais cujo *worfklow* é completamente baseado em sinais *SDI* mas tende a desaparecer à medida que a indústria está a migrar para soluções baseadas unicamente em *IP*.

Para além disso, a vantagem de se seguir o *SMPTE 2022-6* seria a possibilidade de no futuro se implementar o *SMPTE 2022-5* para controlo de erros (*FEC*) durante a transmissão dos pacotes. Ora isto faz sentido para as principais aplicações pensadas pela *SMPTE* quando escreveu esta norma: contribuição e distribuição primária. No entanto, a *MOG Technologies* pretende implementar esta solução de transporte de vídeo sobre *IP* para troca de conteúdos entre equipamentos dentro do estúdio. No estúdio, os equipamentos estão separados por distâncias curtas e os clientes que investem neste tipo de soluções obrigatoriamente devem garantir uma rede estável, isto é, não sujeita a erros.

Assim, a *NMOS* por encapsular apenas o vídeo segundo o *RFC 4175* e não todo o sinal *SDI*, enquadrar-se nos objetivos desta dissertação e ser uma especificação em aberto e precursora que cativou um grande interesse e participação por parte de várias entidades do mundo da televisão, foi a escolhida para a implementação desta fase da dissertação.

3.4.2.1 *NMOS Header*

Como referido em 2.6.6.1, segundo esta especificação devem-se incluir cabeçalhos extra no primeiro e último pacote *RTP* de cada *grain*, que neste caso em que o *flow* é vídeo não comprimido representa um *frame*. Como tal foi desenvolvido um módulo para esse efeito que é chamado entre o submódulo *RTP Buffer* e *RTP sender*.

Este módulo preenche os cabeçalhos extra da seguinte forma:

- ***PTP Sync Timestamp e PTP Origin Timestamp***

  A utilização do *Precision Time Protocol (PTP)* está definida noutra especificação da *NMOS*: a *AMWA NMOS Discovery and Registration Specification (IS-04)* e o mesmo não foi implementado por estar fora do âmbito desta dissertação. O módulo desenvolvido está preparado para caso esta especificação seja implementada, ser facilmente integrada no módulo.

  Assim, o campo referente aos segundos foi preenchido preenchido de acordo com o *Unix Timestamp*, ou seja, o número de segundos decorridos desde 1 de Janeiro de 1970 00:00:00 . O campo referente aos nanosegundos é preenchido com o número de nanosegundos decorridos desde o preenchimento deste valor no primeiro pacote do primeiro *frame* do segundo em questão, voltando a zero ao fim de cada segundo. Por exemplo, num vídeo com 25 *fps* o valor do campo *nanoseconds* do primeiro pacote do primeiro frame seria "0"e o valor deste campo no último pacote do 25º frame seria os nanosegundos que decorreram desde o preenchimento do primeiro pacote. No 26º frame, como já corresponde a outro segundo, o primeiro pacote desse *frame* seria de novo 0 e assim sucessivamente.

  Para o caso de uso desta fase da dissertação, em que só existe uma *Source* a emitir vídeo não comprimido em tempo-real os valores do *PTP Sync Timestamp* e do *PTP Origin Timestamp* vão ser iguais. Só no caso de por exemplo, para além do vídeo a ser recebido em tempo-real se quisessem incluir repetições, estes valores não corresponderiam.

- ***SMPTE ST 12-1 Timecode***

Este valor é preenchido com o tempo decorrido desde o início do vídeo na forma *"horas:minutos:segundos:número do frame"* tal como descrito na norma em questão.

- ***Flow Identifier* e *Source Identifier***

  Para o *Flow Identifier* é gerado um *UUID* novo sempre que se envia um novo *flow* de vídeo. O *Source Identifier* tem sempre o mesmo *UUID* pois nesta dissertação, como já referido, a *Source* é sempre a mesma.

- ***Grain Duration***

  Este campo é preenchido simplesmente com o valor "1/*frame rate*"do vídeo a transmitir.

3.4.2.2 *SDP UHD NMOS*

O ficheiro *SDP* [32] a enviar ao recetor antes de se enviarem os pacotes de vídeo sobre *RTP* foi preenchido conforme mostrado no exemplo a seguir:

1. **"v=0"**

   Esta linha indica que a versão da especificação *SDP* seguida é a 0 pois é a referente à descrita em *RFC 4566*.

2. **"o=- 1443716955 1443716955 IN IP4 192.168.1.102"**

   O formato desta linha é "*o=<username> <sess-id> <sess-version> <nettype> <addrtype> <unicast-address>*". O primeiro campo referente ao *<username>* é arbitrário uma vez que não é relevante para a sessão, pelo que se optou pelo caractere *"-"*. Os campos *<sess-id>* e *<sess-version>* são *strings* numéricas que indicam respetivamente o *ID* da sessão e o *ID* da versão que varia caso a sessão seja alterada. Para estes dois campos o *ID* é conseguido através de um *timestamp* que indica os segundos decorridos desde *00:00:00 (UTC)* de 1 Janeiro de 1970 até ao envio do 1º pacote. De seguida, identifica-se que o tipo de rede é internet (*"IN"*), será utilizada a quarta versão do protocolo *IP* (*"IP4"* e o endereço *IP* do servidor pelo qual o vídeo será enviado é *"192.168.1.102"*.

3. **"s=NMOS Stream"**

   Indica o nome em texto da sessão.

4. **"t= 0 0"**

   Indica o tmepo de ínicio e fim da sessão no formato *"t=<star-time> <end-time>*. Ambos os campos são preenchidos a zero pois a sessão é permanente. Isto é, inicia no instante em que se deseja começar a enviar o vídeo e a data do seu fim não está definida.

5. **"m=video 2000 RTP/AVP 96"**

   Nesta linha é identificado que o conteúdo a enviar é vídeo e que este será enviado pela porta 2000. De seguida é identificado que será utilizado o protocolo *RTP* e que o *payload* do

vídeo tem o perfil 96 descrito no *RTP Profile for Audio and Video Conference With Minimal Control (AVP)* [31]. Este perfil corresponde a um perfil dinâmico que é o caso do descrito no *RFC 4175* utilizado.

6. **"c=IN IP4 192.168.1.101"**

   Começa-se por identificar que o tipo de rede a utilizar é *Internet* e o protocolo *IP* é a quarta versão do protocolo *IP*, o *IPv4*. O último parâmetro indica o endereço de receção.

7. **"a=source-filter:incl IN IP4 192.168.1.101 192.168.1.102"**

   O primeiro dos atributos definidos neste *SDP* é o *source-filter* [43]. O objetivo do *source-filter* é ajudar a proteger o recetor de tráfego enviado de fontes ilegítimas. Assim, neste campo é indicado que o endereço *192.168.1.101* deve aceitar apenas pacotes enviados sobre uma rede *IP* utilizando o protocolo *IPv4* pelo endereço *192.168.1.102*.

8. **"a=rtpmap:96 raw/90000"**

   Indica que o tipo de *payload* dos pacotes tem o perfil 96, que corresponde a um *payload* dinâmico referente a um formato não comprimido e a frequência do *timestamp* dos pacotes *RTP* é de 90000 *Hz*.

9. **"a=fmtp:96 sampling=YCbCr-4:2:2; width=3840; height=2160; depth=10; colorimetry BT709-2; interlace=0"**

   Este campo é utilizado para definir parâmetros específicos do vídeo a enviar. Neste exemplo seria enviado vídeo *UHDTV1* com espaço de cor *YCbCr* e subamostragem de cor *4:2:2*, cada píxel é representado por 10 *bits* e o espaço de cor é o *BT709-2*.

10. **"a=extmap:1 urn:x-nmos:rtp-hdrext:sync-timestamp**

    a=extmap:2 urn:x-nmos:rtp-hdrext:origin-timestamp

    a=extmap:3 urn:ietf:params:rtp-hdrext:smpte-tc 3600@90000/25

    a=extmap:4 urn:x-nmos:rtp-hdrext:flow-id

    a=extmap:5 urn:x-nmos:rtp-hdrext:grain-flags

    a=extmap:6 urn:x-nmos:rtp-hdrext:grain-duration

    a=extmap:7 urn:x-nmos:rtp-hdrext:grain-flags"

    Nestas linhas é indicada a ordem pela qual os *header extensions* específicados na *NMOS* são enviados no primeiro pacote *RTP* de cada *frame* de acordo com a norma *RFC 5285- A general mechanism for RTP Header Extensions*. Neste caso a ordem dos *header extensions* é: *PTP Sync Timestamp, PTP Origin Timestamp, SMPTE ST 12-1 (SMPTE 12M) Timecodes, Flow Identifier, Source Identifier, Grain Duration, Grain Flags.*

    Os atributos presentes na linha referente ao *Timecode SMPTE 12M* estão definidos na norma *RFC5484: Associating Time-Codes with RTP Streams* [44], tendo a linha o seguinte formato: *a=extmap:<id> urn:ietf:params:rtp-hdrext:smpte-tc <dur>@<rate>/<tc base>*.

O campo <*rate*> diz respeito à frequência do *timestamp* dos pacotes *RTP* , o valor 90000 está definido no *RFC 4175*. O parâmetro <*tc base*> é de 25 uma vez que a *frame rate* do vídeo a enviar é de 25 *fps* neste caso. Assim, a duração do *frame* é de 90000/25, ou seja 3600, que corresponde ao número de *"ticks"* do relógio para cada *frame*.

3.4.3 Integração no *mxfSPEEDRAIL*

Para integrar os módulos desenvolvidos no *mxfSPEEDRAIL* (solução *sCapture*) dividiu-se o trabalho em diversas fases:

1. Captura de vídeo gerado pelo *Signal Generator* e envio sobre *RTP*.
2. Leitura do vídeo recebido sobre *RTP*
3. Implementação segundo a *NMOS*.
4. Captura de vídeo através do módulo de captura desenvolvido na primeira fase da dissertação e envio sobre *RTP* segundo a *NMOS*.

3.4.3.1 Captura de vídeo gerado pelo *Signal Generator* e envio sobre *RTP*

Uma vez que o *mxfSPEEDRAIL* não utiliza a biblioteca *OpenCV* utilizada em 3.4.1 mas sim funções e formatos de dados próprios para manipulação de vídeo, esta fase teve como primeiro objetivo perceber como realizar essa manipulação com as ferramentas da empresa.

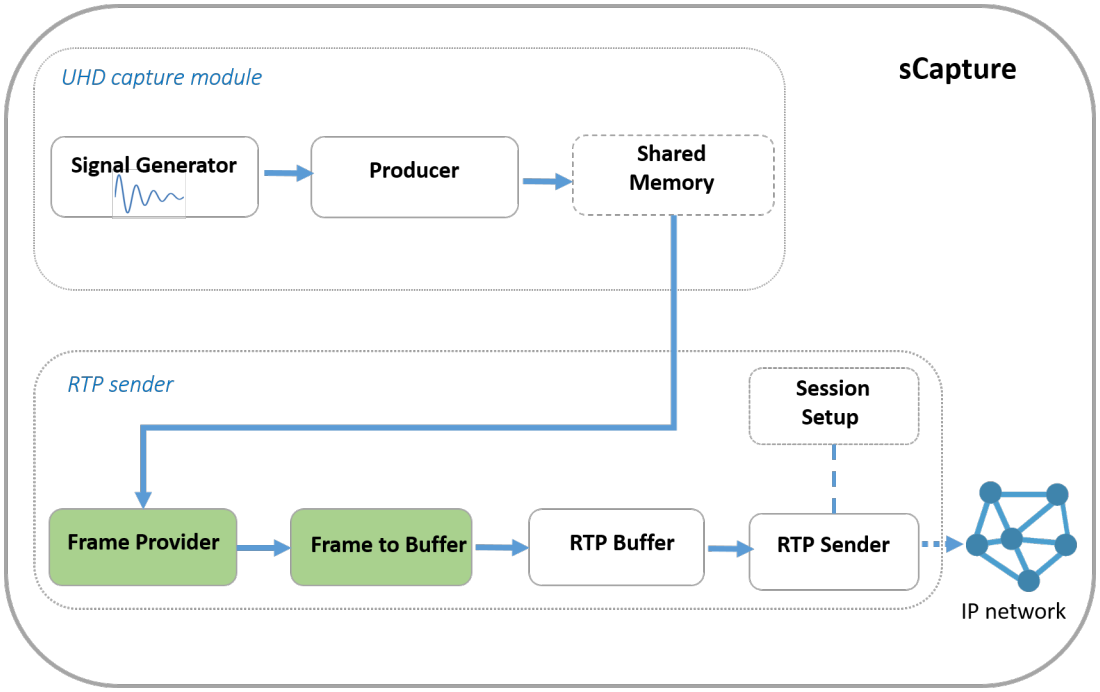

Figura 3.11: Diagrama do *workflow* da fase de integração do módulo *RTP sender* no *mxfSPEEDRAIL* com vídeo gerado pelo *Signal Generator*

Aproveitou-se o trabalho desenvolvido na primeira fase da dissertação referente ao módulo de de captura de vídeo de *UHD* para gerar o sinal através do *Signal Generator* e escrevê-lo na memória partilhada, *Shared Memory*, através do *Producer*.

De seguida, adaptou-se o módulo *RTP sender* já explicado em 3.4.1.1 para poder ler vídeo da *Shared Memory* e enviá-lo sobre *RTP*. Para isso, teve que se adaptar o módulo *Frame Provider* que na "solução simples de transporte de vídeo sobre *RTP*"lia um ficheiro de vídeo presente em disco *frame* a *frame* através das funções do *OpenCV* e disponibilizáva-o ao módulo seguinte. O módulo *Frame to buffer* também precisou de ser adaptado uma vez que o seu *input* modou de formato, no entanto a sua função manteve-se: receber um *frame* completo de vídeo e manipulá-lo para um *buffer* de uma só linha.

3.4.3.2 Leitura do vídeo recebido sobre *RTP*

Nesta fase desenvolveu-se um módulo de receção de vídeo sobre *RTP* no *sCapture*. Para isso bastou adaptar o submódulo *"Buffer to Frame"* desenvolvido em 3.4.1.2 para o formato de dados do *mxfSPEEDRAIL*. Escolheu-se que este submódulo apenas guardasse os *frames* recebidos em disco uma vez que o módulo de receção não era o foco da dissertação e serviu apenas para efeitos de testes. Não obstante, poderia-se facilmente adaptar este módulo para integração com o *Producer* e o restante *worfklow* do *sCapture*.

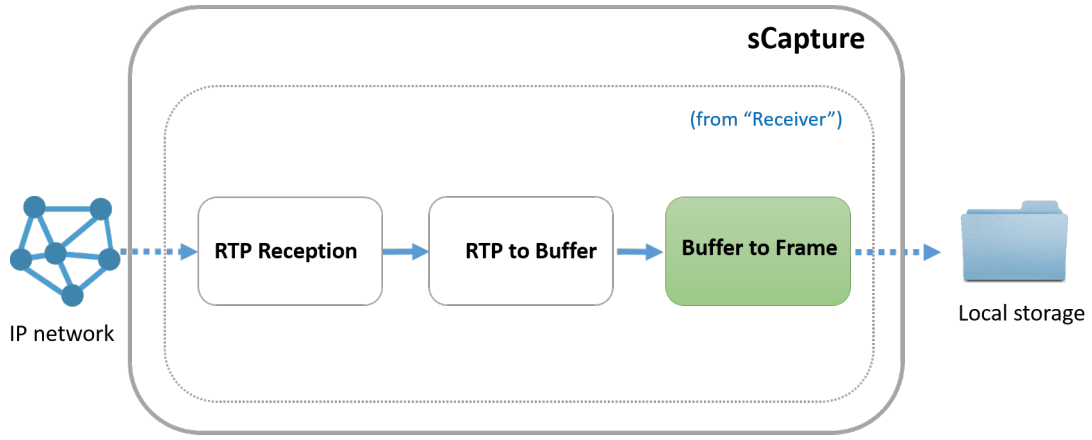

Figura 3.12: Diagrama do *workflow* do módulo *RTP receiver* adaptado para receber o vídeo enviado sobre *RTP* no *mxfSPEEDRAIL*

Depois deste módulo estar concluído utilizaram-se os dois servidores que antes enviavam e recebiam vídeo de *UHD* por *SDI*, para enviar e receber vídeo de *UHD* sobre *RTP*. Para isso, instalou-se em ambos o *sCapture* e ligou-se os servidores à rede através de uma *interface* de 10 *GbE*. Assim, com um servidor a enviar o vídeo sobre *RTP*, correndo o módulo de envio explicado

na secção anterior e outro a correr o módulo de receção explicado nesta secção, foi possível avaliar o sucesso da implementação.

3.4.3.3 Implementação segundo a *NMOS*

Nesta fase começou-se por integrar os módulos desenvolvidos também na fase anterior para preenchimento e leitura dos cabeçalhos segundo a especificação *NMOS In-stream Identity and Timing Specification* e para isso não foi necessária nenhuma adaptação dos mesmos. De seguida, de forma a conseguir transferir o ficheiro *SDP* entre o emissor e o recetor antes da transmissão dos pacotes *RTP*, começou-se por adicionar a ambos os submódulos *Session Setup* dos módulos *RTP sender* e *RTP receiver* a funcionalidade de escrever e ler esse ficheiro, respetivamente. Seguidamente, adicionou-se a possibilidade de através do servidor *Web* utilizado pelo *Controller* para configurar e controlar os módulos do *sCapture* ser possível transferir esse ficheiro entre os módulos.

3.4.3.4 Captura de vídeo através do módulo de captura desenvolvido na Fase 1 da dissertação e envio sobre *RTP* segundo a *NMOS*

Depois de garantir que todo o *workflow* do lado do envio de vídeo sobre *RTP* funcionava com vídeo gerado pelo *Signal Generator*, se implementar um módulo de receção noutra máquina, ligar ambas à rede para avaliar o sucesso da transmissão e se implementar o envio e a receção segundo a *NMOS*, foi possível testar todo o *workflow* do objetivo da dissertação: capturar vídeo de *UHD* encapsulado em *SDI* e enviá-lo sobre *RTP*. Para isso, comparativamente ao *wofklow* ilustrado na Fig 3.11 foi apenas necessário alterar o *Producer* para que em vez de escrever na memória partilhada vídeo gerado pelo *Signal Generator*, escrevesse o vídeo recebido pela *DeckLink 4k Pro* gerado pelo *rPlayer* instalado em outra máquina, tal como já tinha acontecido em 3.3.3.

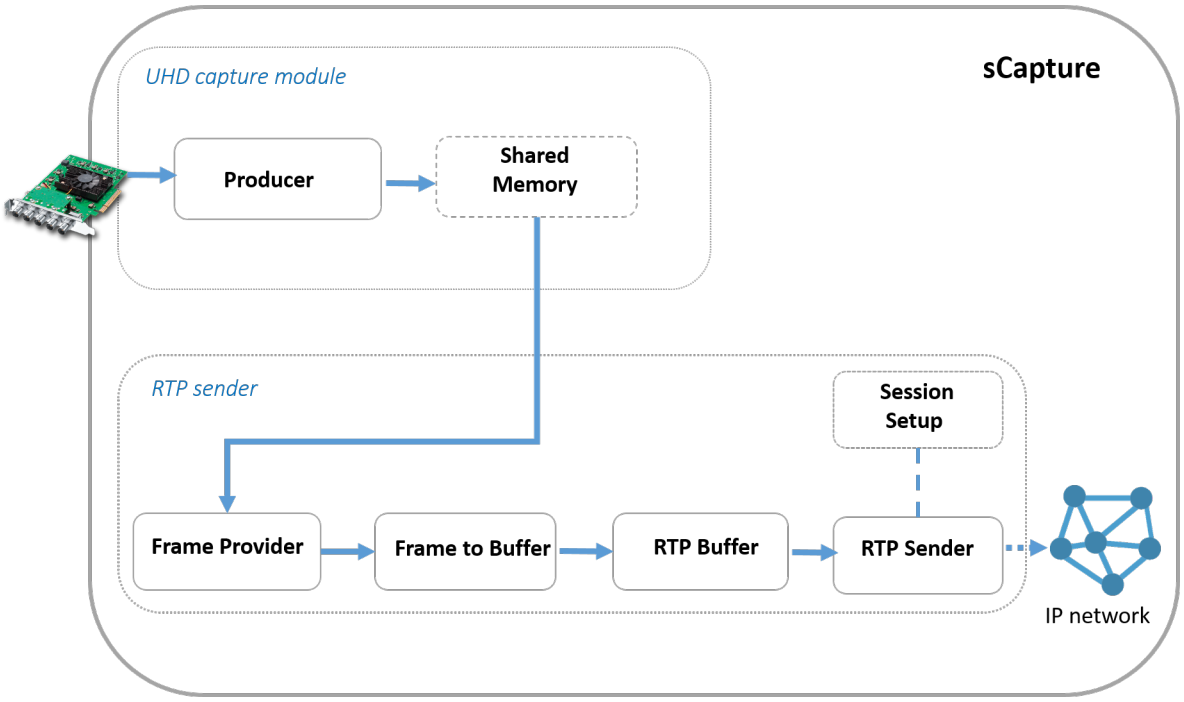

Figura 3.13: Diagrama do *Worklow* da fase de captura de vídeo por *SDI* e envio do mesmo sobre *RTP*.

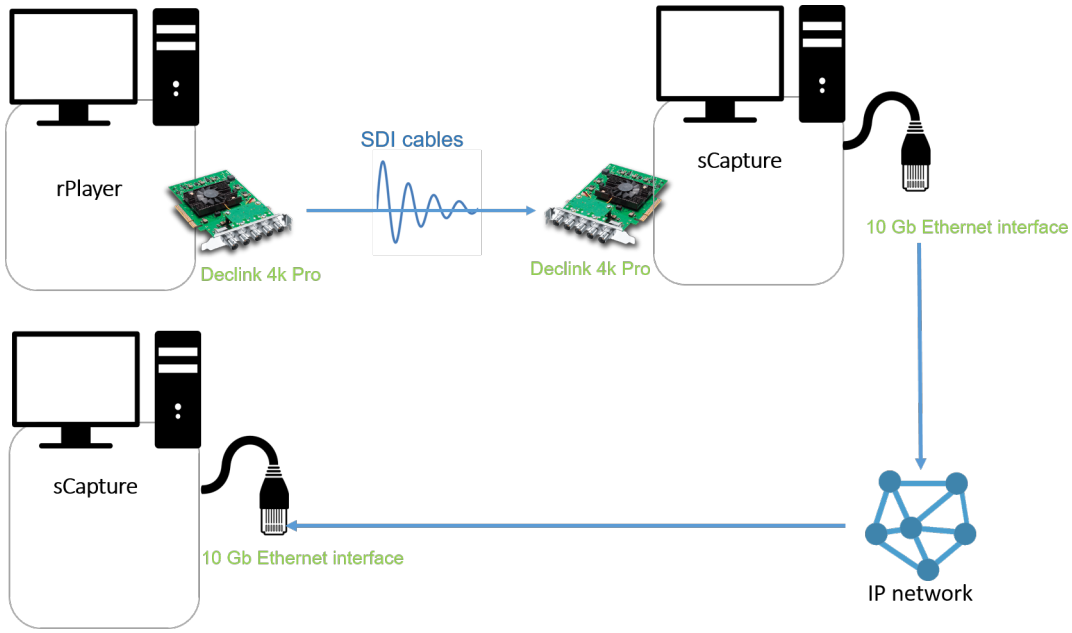

Figura 3.14: Diagrama geral do *workflow* desta fase.

Capítulo 4

Resultados obtidos

4.1 Resultados da Fase 1: Captura de vídeo *UHD* por *SDI*

Nesta fase os resultados obtidos foram sobretudo avaliados de forma visual e pela existência de *frames* perdidos ou corrompidos através de testes unitários abrangendo todos os tipos de vídeo de *UHD* implementados.

Na Subfase 1: Implementação *Signal Generator* e *Producer* os resultados obtidos foram avaliados de forma visual através do *local monitor*. Todos os formatos desenvolvidos foram implementados com sucesso.

Durante a Subfase 2: Integração com o módulo *Processor (wrap/transcode)* os resultados foram obtidos também de forma visual, lendo o vídeo presente em disco através de um *media player*, tendo-se verificado o sucesso da implementação desta fase.

Para avaliar os resultados da Subfase 3: Integração com a placa de captura *DeckLink 4k Pro* recorreu-se ao servidor *web* com *interface* gráfica desenvolvido pela *MOG Technologies* para efeitos de testes dos módulos do *mxfSPEEDRAIL* tanto do lado de *playback*, para configurar os módulos do *rPlayer* e visualizar o vídeo a enviar, como do lado da captura, para configurar os módulos do *sCapture* e visualizar o vídeo recebido.

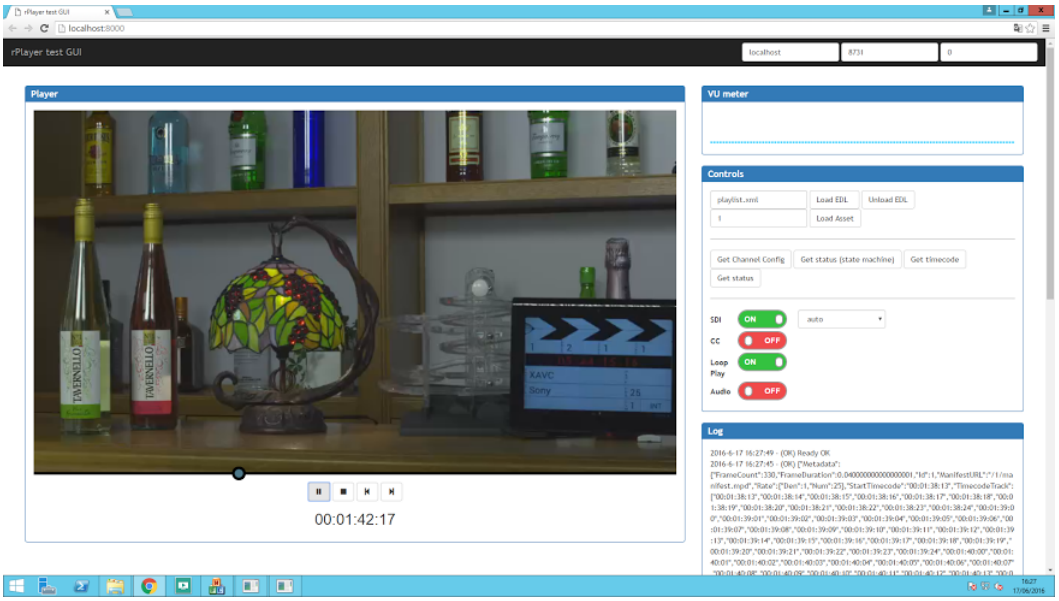

Figura 4.1: *Interface* de testes do *rPlayer* durante o envio de vídeo *UHDTV1* por *SDI*.

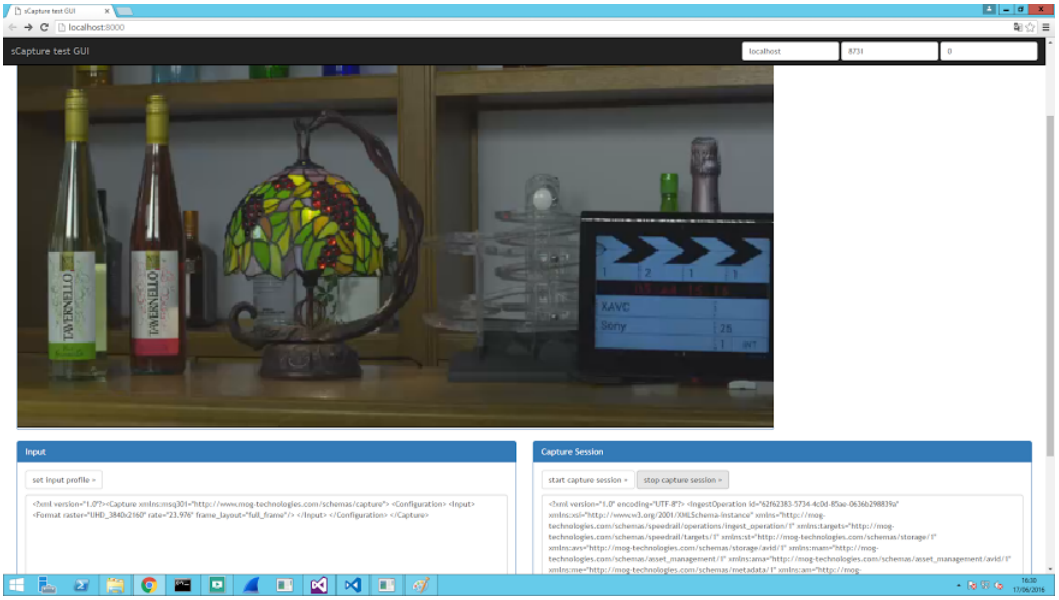

Figura 4.2: *Interface* de testes do *sCapture* durante a captura de vídeo *UHDTV1* por *SDI*.

Para verificar a existência de *frames* perdidos recorreu-se ao *software Media Express* da *Blackmagic* para a captura e visualização do vídeo gerado. Este *software*, instalado no servidor com o papel de recetor, avisa caso detete a existência de *frames* perdidos, o que se verificou nesta fase. Para descobrir as causas foi feito um teste com o *software Signal Generator* da *Blackmagic*. Este *software* foi instalado do lado do emissor e gera e envia sobre *SDI* vídeo de todos os formatos suportados pela placa *DeckLink 4k Pro*. Mesmo com este *Software* oficial do fabricante da placa,

verificou-se a existência de *frames* perdidos, o que leva a crer que as limitações são do *Hardware* dos servidores utilizados e não dos módulos desenvolvidos.

Foi também medida a utilização do *CPU* e da memória *RAM* durante esta subfase em ambos os servidores e concluiu-se que não estavam a ser completamente utilizados, pelo que não estavam a limitar o sucesso da implementação. Estas medições podem ser vistas nas Figuras 4.3 e 4.4, referentes ao envio de vídeo *UHDTV1* a 25 *fps*.

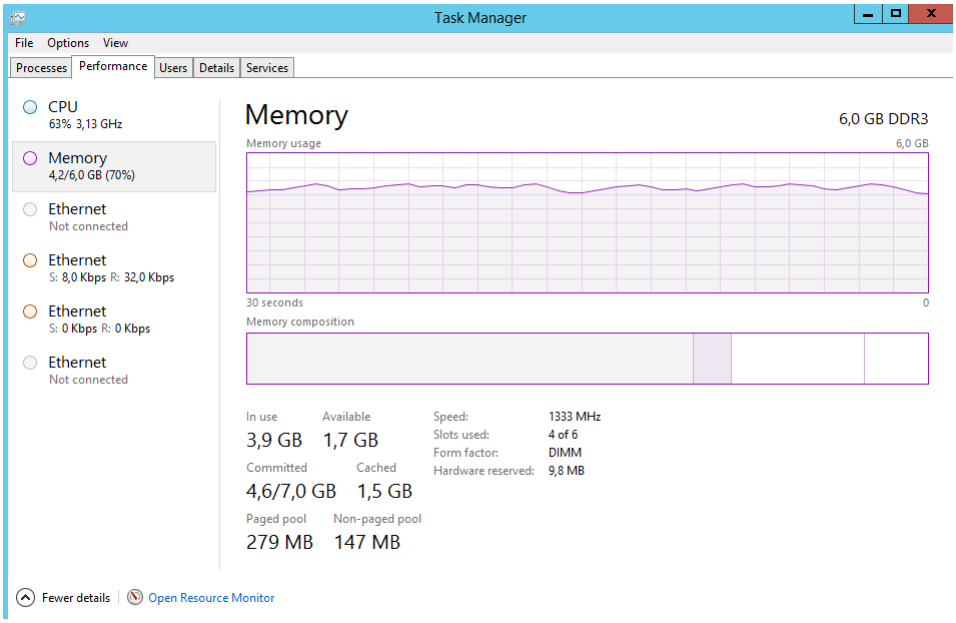

Figura 4.3: Utilização do *CPU* e memória *RAM* no servidor onde foi implementado o *sCapture*, a capturar via *SDI* vídeo *UHDTV1* a 25 *fps*.

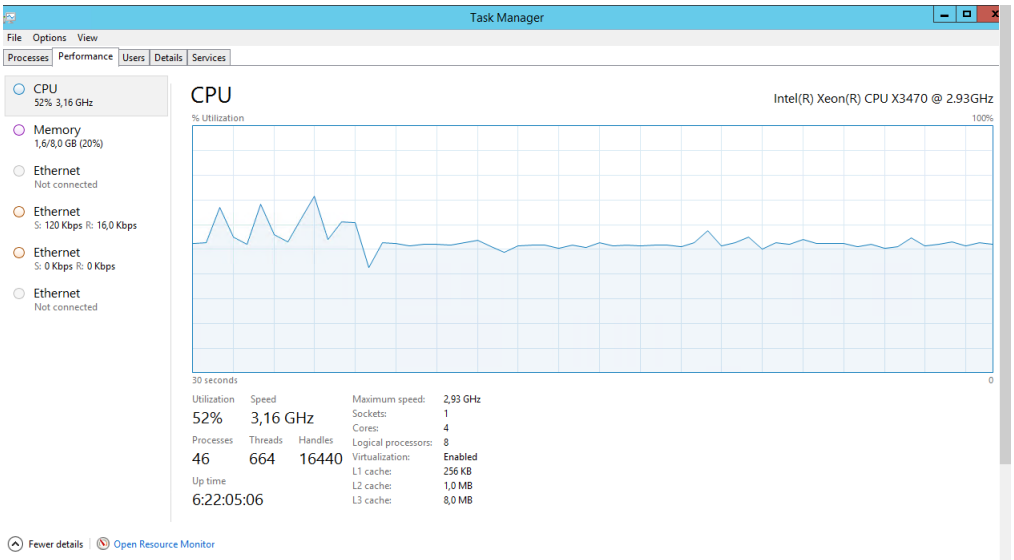

Figura 4.4: Utilização do *CPU* e memória *RAM* no servidor onde foi implementado o *rPlayer*, a transmitir por *SDI* vídeo *UHDTV1* a 25 *fps*.

A limitação em causa concluiu-se ser a velocidade do disco de ambos os servidores, emissor e recetor. Esta limitação foi confirmada recorrendo ao *software* da *Blackmagic Blackmagic Disk Speed Test* cujos resultados podem ser vistos na Figura 4.5 e Figura 4.6. Para o emissor enviar e o recetor receber vídeo de *UHD* precisaria de uma velocidade de escrita e leitura em disco maior que 375 *MB/s* de acordo com os *payloads* apresentados na Figura 2.14. No entanto verifica-se que a velocidade dos discos dos servidores utilizados é cerca de 10 vezes menor que esse valor.

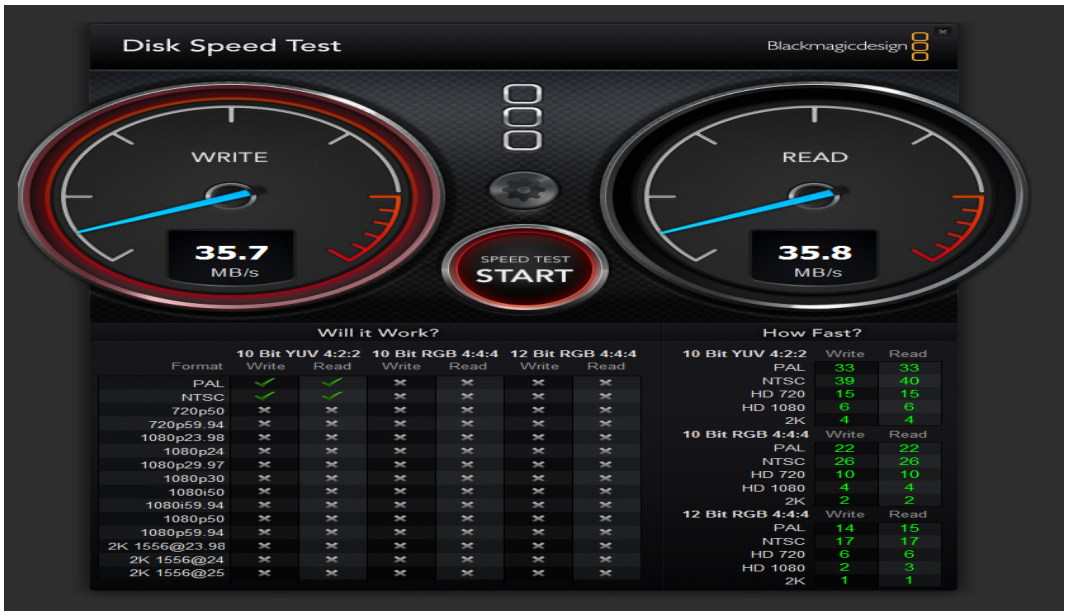

Figura 4.5: Teste de velocidade do disco do lado do recetor com o *software Blackmagic Disk Speed Test*

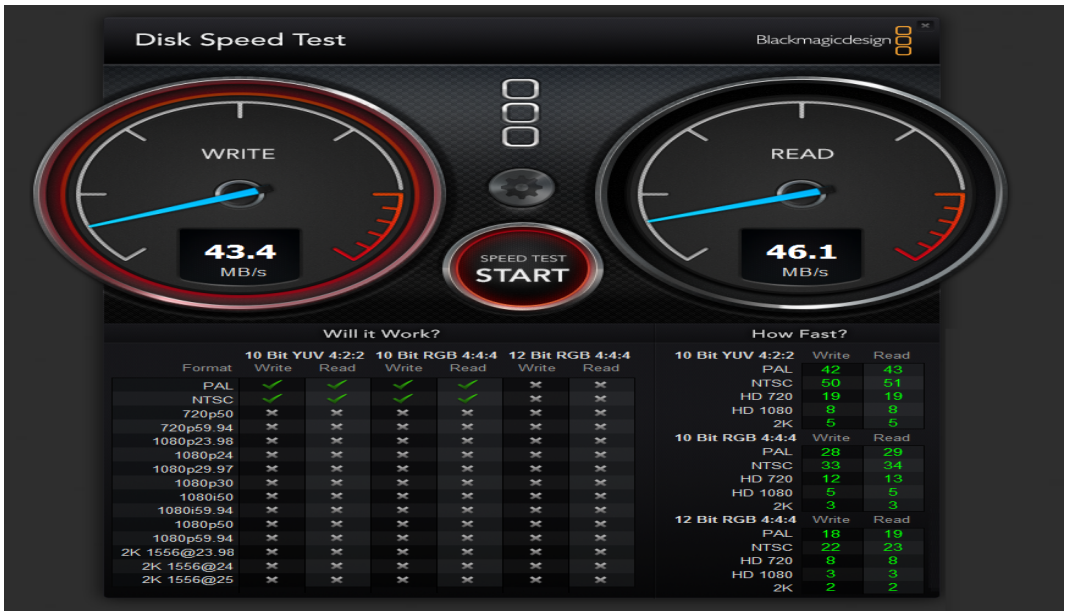

Figura 4.6: Teste de velocidade do disco do lado do emissor com o *software Blackmagic Disk Speed Test*

4.2 Resultados da Fase 2: Envio de vídeo de *UHD* sobre *RTP*

Na primeira subfase Desenvolvimento de uma solução simples de transporte de vídeo sobre *RTP* os módulos de envio e receção de vídeo sobre *RTP* estavam a correr no mesmo computador logo a rede não era um fator a considerar. Assim, apenas se verificou visualmente que o vídeo era enviado e recebido corretamente recorrendo à *GUI* disponibilizada pela *OpenCV*.

Depois na subfase Implementação segundo a *NMOS* avaliou-se o sucesso da implementação primariamente verificando que as extensões de cabeçalho definidas no *NMOS* tinham sido implementadas e recebidas corretamente. Para isso, recorreu-se a *breakpoints* do lado do *Sender* de forma a que quando estes cabeçalhos extra são introduzidos, isto é, no primeiro e no último pacote de cada *frame* se pudesse parar o envio de pacotes *RTP* e anotar os valores presentes nos cabeçalhos extra dos mesmos. No lado do *Receiver* foram lidos todos os cabeçalhos desses pacotes individualmente através de funções desenvolvidas para o efeito, verificando-se que os mesmos correspondiam aos cabeçalhos enviados. Assim, pôde-se concluir que tanto a função de preenchimento dos cabeçalhos como as funções de leitura dos mesmos estavam corretamente implementadas.

Ao integrar-se os módulos de envio e de receção no *Speedrail* para além da avaliação visual da implementação foi importante avaliar aspetos relacionados com a rede.

Para calcular o tempo médio de atraso na rede criou-se um módulo de envio de pacotes sobre *RTP* para esse efeito. Este módulo enviava pacotes com um *payload* constante de 1350 *bytes*, que é o número médio de *bytes* contidos nos pacotes desta fase que efetivamente transportam vídeo, presente num *buffer* previamente preenchido para reduzir ao mínimo o tempo de processamento. Desenvolveu-se também um módulo de receção que apenas recebia os pacotes, sem os processar e media a diferença de tempo entre a receção de pacotes sucessivos.

Para este teste enviaram-se 15360 pacotes por ser o o número de pacotes necessários para enviar vídeo segundo a *NMOS* no formato *UHDTV1* a 25 *fps* com subamostragem de cor *4:2:2* e 10 *bits* de profundidade. As medições foram feitas do lado da receção.

Tabela 4.1: Atrasos medidos na rede

Bytes por pacote	1350
Número de pacotes enviados	15360
Delay minímo entre pacotes	109 *ns*
Delay máximo entre pacotes	61330758 *ns*
*V*ariação máxima do *delay*	61330649 *ns*
Delay médio entre pacotes	5576 *ns*

Assim, para enviar por exemplo um *frame* de vídeo de *UHD* com as características referidas, uma vez que são necessários 15360 pacotes por *frame*, cada *frame* demoraria em média 86 milisegundos. Isto, neste caso em que o processamento dos pacotes é mínimo no lado do emissor uma vez que o seu *payload* está previamente preenchido num *buffer*. Este resultado não é satisfatório

pois com este *delay* médio na rede só seria possível serem transmitidos no máximo 10 *frames* completos por segundo, o que é obviamente muito baixo.

De seguida mediu-se o tempo de processamento dos pacotes para o envio de vídeo sobre *RTP* de todo o módulo *RTP sender*. O tempo de processamento de cada *frame* medido por este módulo integral foi de 141,225 *ms*. No caso de um *frame* do vídeo considerado seria apenas possível enviar no máximo também apenas 7 *frames* completos por segundo, isto numa rede sem atrasos o que já se verificou que não é o caso.

Finalmente mediu-se o atraso total entre *frames* considerando o atraso medido na rede e o atraso devido ao processamento. O valor médio expectável seria a soma dos dois atrasos medidos, isto é a soma de 86 *ms* com 141*ms* que é igual a 227 *ms*. O atraso médio medido entre os pacotes foi de 168,25 *ms*, ligeiramente mais baixo do que o valor expectável. Esta diferença de valores deve-se principalmente ao facto de se ter considerado o valor do *delay* médio entre pacotes na rede e a variação entre o valor máximo e mínimo ser muito grande. Assim, com um atraso de 168,25 *ms* é possível enviar 5 *frames* completos de vídeo sobre *RTP* segundo a *NMOS* no formato de *UHDTV1* com subamostragem de cor *4:2:2* e 10 *bits* de profundidade.

Estes valores foram confirmados recorrendo-se ao "Gestor de Tarefas"do Windows. Enviando-se cerca de 5 *frames* por segundo de vídeo como confirmado pelos resultados anteriores, o tráfego na rede era de cerca de 830 Mbs como ilustrado nas duas Figuras seguintes

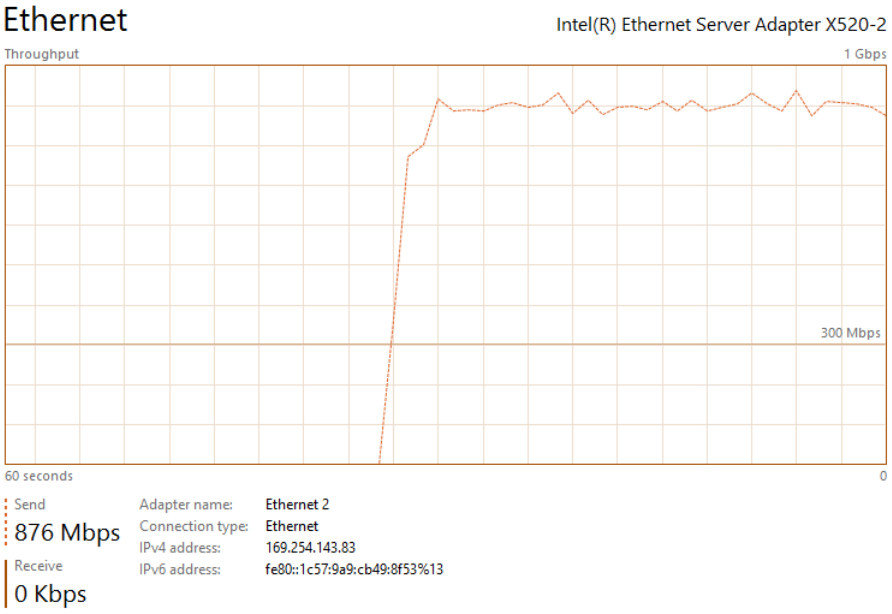

Figura 4.7: Tráfego na rede do lado do emissor a enviar vídeo sobre *RTP* segundo a *NMOS* no formato de *UHDTV1* com subamostragem de cor *4:2:2* e 10 *bits* de profundidade.

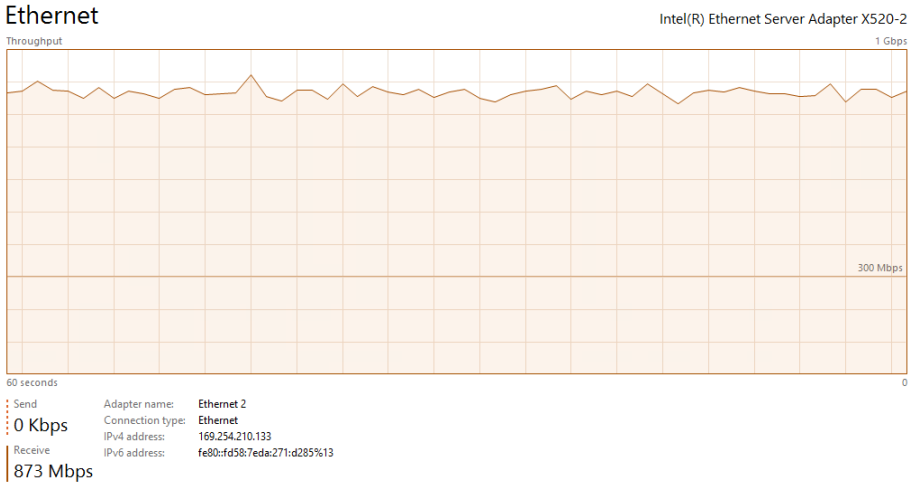

Figura 4.8: *T*ráfego na rede do lado do recetor a receber vídeo sobre *RTP* segundo a *NMOS* no formato de *UHDTV1* com subamostragem de cor *4:2:2* e 10 *bits* de profundidade.

Verificou-se ainda que existia perda de pacotes numa média 8 pacotes perdidos em cada 3840000. No caso do formato considerado neste subcapítulo a 25 *fps* corresponde a 8 pacotes perdidos em cada 10 segundos do vídeo.

Capítulo 5

Conclusões e Trabalho Futuro

Neste capítulo são apresentadas as principais conclusões do trabalho realizado nesta dissertação, avaliando o cumprimento e satisfação dos objetivos definidos para a mesma. Para além disso, apresenta-se algumas considerações sobre a sua contribuição no âmbito da produção de televisão bem como algum possível trabalho futuro que poderá ser desenvolvido partindo das implementações realizadas.

5.1 Conclusões

O trabalho desenvolvido cumpriu os objetivos iniciais da dissertação embora a sua implementação apresente algumas limitações tecnológicas impostas pelos equipamentos utilizados já apresentadas no capítulo anterior.

Na primeira parte da dissertação, referente à captura de vídeo de *UHD* tudo indica que o módulo de *software* desenvolvido funcionaria corretamente em *hardware* adequado. Em primeiro lugar, porque os diversos testes unitários realizados nas fases antecessoras à da integração com a placa de captura *Decklink 4k Pro* passaram com sucesso. Em segundo lugar, porque na fase em que se integrou a placa de captura na solução desenvolvida e se verificou que eram perdidos *frames*, foram realizados testes substituindo esta solução por *software* oficial da *Blackmagic* e acontecia o mesmo.

Na segunda parte da dissertação, referente ao envio do vídeo capturado sobre *IP* os atrasos devido ao processamento também demonstraram a existência de limitações tecnológicas do *Hardware* utilizado para o sucesso do módulo. Para além disso, embora a *interface* de rede utilizada fosse adequada verificaram-se atrasos na rede e perda de pacotes mesmo que residual.

Conclui-se que a adoção da *NMOS* foi uma boa escolha pois minimiza os atrasos existentes uma vez que são necessários menos pacotes do que nos seus "concorrentes". Além do mais, por ser uma especificação aberta com cada vez mais entidades a contribuir para a mesma demonstra ter potencial para se assumir durante os próximos tempos como uma norma global para as soluções de televisão baseadas em *IP*, eliminando o problema atual da interoperabilidade entre diferentes equipamentos de diferentes fabricantes.

Conclui-se também que a produção de televisão de *UHD* implica grande capacidade de processamento e por isso os equipamentos a utilizar devem cumprir os requisitos necessários para lidar com estes formatos. Estes requisitos implicam memória de *CPU* e *RAM* adequados ao tipo de vídeo a transmitir bem como velocidade de escrita e leitura em disco adequada.

Foi também possível constatar-se na prática que as soluções de produção de televisão baseadas em *IP* beneficiam das vantagens referidas no estado de arte. Para além do *hardware* ser mais barato, a flexibilidade e escalabilidade destas infraestruturas é muito maior. Por exemplo, caso se pretenda utilizar uma placa de captura de um outro fabricante que não seja a *Blackmagic*, seria necessário substituir o módulo onde são adaptados os formatos utilizados pelo *Blackmagic SDK 10.4.1* aos formatos internos do *mxfSPEEDRAIL*. Se a captura em vez de por uma *interface SDI* fosse recebida sobre *IP* não seria necessária qualquer adaptação desde que obviamente se utilizassem os mesmos protocolos e/ou especificações. Para além disso, caso surja um novo formato de vídeo será mais fácil adaptar o módulo de envio sobre *IP* desenvolvido do que o módulo de captura de vídeo por *SDI*. No primeiro módulo, caso sejam seguidas as mesmas especificações não é necessário qualquer alteração ao módulo desenvolvido podendo ser necessário no máximo uma alteração da *interface* física de rede. Por outro lado, no segundo seria necessário utilizar outra versão do *SDK* e utilizar outra placa de captura, o que também implicaria alterações no módulo.

5.2 Trabalho Futuro

Como trabalho futuro prevê-se em primeiro lugar integrar ambas as soluções desenvolvidas em servidores com melhor *hardware* e avaliar o seu desempenho.

Deve-se também no futuro testar o módulo desenvolvido na primeira parte da dissertação, o *UHD capture module*, utilizando uma placa de captura que suporte vídeo no formato *UHD2*, o que não acontecia com a *Declink 4k Pro*.

Quanto ao módulo *RTP sender* deve-se adaptar o módulo caso haja alterações na especificação *In-stream Signaling of Identity and Timing information for RTP streams* ainda em desenvolvimento à data de escrita desta dissertação e em conjunto com os outros participantes da *NMOS* testar a solução desenvolvida. Para além disso, outro trabalho será integrar este módulo numa solução de produção de televisão que implemente a outra especificação da *NMOS*, a *AMWA NMOS Discovery and Registration Specification (IS-04)* para conseguir fazer uso dos *timestamps* sincronizados entre os diferentes elementos da rede segundo o *Precision Time Protocol*.

Por fim, o módulo *RTP receiver* desenvolvido nesta dissertação para efeitos de testes envolvendo o *RTP sender*, deverá ser desenvolvido completamente num futuro próximo para que o *mxfSPEEDRAIL* permita não só enviar como também receber vídeo por *RTP*.

Referências

[1] J. Wilkinson e B. Devlin. N. Wells, O. Morgan. *The Mxf Book: An Introduction to the Material Exchange Format*, 2006. URL: http://books.google.pt/books?id=6P9fMQEACAAJ.

[2] HD Video Pro. *Formats Explained*. Disponível em: 2016-06-10. URL: http://www.hdvideopro.com/columns/help-desk/formats-explained/page-2.

[3] Sareesh Sudhakaran. *Deconstructing RAW*. Disponível em: 2016-06-10. URL: ttp://wolfcrow.com/blog/deconstructing-raw-part-i.

[4] Society of Motion Picture e Television Engineers. Smpte uhdtv ecosystem study group report, Março 2014.

[5] *Deltacast. 4k Video*. Disponível em: 2016-06-10. URL: http://www.deltacast.tv/technologies/4k-video.

[6] *Society of Motion Picture and Television Engineers. UHD in a Hybrid SDI/IP World*. SMPTE MONTHLY EDUCATION WEBCAST SERIES, Novembro 2015.

[7] The Internet Engineering Task Force (IETF). *RFC3550. RTP: A Transport Protocol for Real-Time Applications*, 2003. URL: URL:https://tools.ietf.org/html/rfc1899.

[8] Mathias Laabs. *SDI over IP*. Disponível em: 2016-06-10. URL: https://www.smpte.org/publications/past-issues/January-2015.

[9] The Internet Engineering Task Force (IETF). *https://tools.ietf.org/html/rfc4175*, 2005. URL: URL:https://tools.ietf.org/html/rfc1899.

[10] Advanced Media Workflow Association. *NMOS Technical Overview* . Disponível em: 2016-06-10. URL: https://github.com/AMWA-TV/nmos/blob/master/NMOS%20Technical%20Overview.md.

[11] *Blackmagic Design. Declink 4k Pro Tech Specs*. Disponível em: 2016-06-15. URL: URL:https://www.blackmagicdesign.com/products/decklink/techspecs/W-DLK-26.

[12] FUNTTEL. Projeto sistema brasileiro de televisão digital: Modelo de implantação os40539, 2004. URL: http://docplayer.com.br/6982917-Cadeia-de-valor-funttel-projeto-sistema-brasileiro-de-televisao-dig
html.

[13] A. N. P. Alves. Interfaces tácteis para ambientes de pós-produção de televisão. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2013.

[14] *Transmission of television and sound programme signal for contribution, primary distribution and secondary distribution*. http://www.itu.int/en/ITU-T/studygroups/2013-2016/09/Pages/q1.aspx. Disponível em: 2016-06-01.

[15] Bastos Rui J. C. Desenvolvimento de sistemas de controlo de equipamento de produção de tv. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2009.

[16] John Hudson. *3Gb/s SDI for Transport of 1080p50/60, 3D, UHDTV1 / 4k and Beyond* , 2013. Disponível em: 2016-06-10. URL: https://www.smpte.org/sites/default/files/2013-09-10-3GSDI-Hudson-V3-Handout.pdf.

[17] John Hudson. *A Quick Tour of Wrappers and MXF*, 2013. Disponível em: 2016-06-10. URL: https://www.smpte.org/sites/default/files/2013-09-10-3GSDI-Hudson-V3-Handout.pdf.

[18] *International Telecommunication Union. The present state of ultra-high definition television*, Julho 2015. URL: http://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-BT.2246-5-2015-PDF-E.pdf.

[19] *SMPTE OV 2036*. Disponível em: 2016-06-10. URL: http://www.hdvideopro.com/columns/help-desk/formats-explained/page-2.

[20] *Migrating SDI media facilities to ip with Smpte 2022-6*. Disponível em: 2016-06-10. URL: http://www.hdvideopro.com/columns/help-desk/formats-explained/page-2.

[21] *Society of Motion Picture and Television Engineers. SMPTE 291M: Ancillary Data Packet and Space Formatting*, 1998.

[22] Micheal Goldman. *What's the best IP forward*. Disponível em: 2016-06-10. URL: https://www.smpte.org/publications/past-issues/January-2015.

[23] Society of Motion Picture e Television Engineers. *SMPTE 2022-6: Transport of High Bit Rate Media Signals over IP Networks (HBRMT)*, 2012.

[24] The Internet Engineering Task Force (IETF). *RFC1899. RTP: A Transport Protocol for Real-Time Applications*, 1996. URL: URL:https://tools.ietf.org/html/rfc1899.

[25] The Internet Engineering Task Force (IETF). *RFC 793. TCP: Transmission Control Protocol*, 1981. URL: URL:https://tools.ietf.org/html/rfc793.

[26] The Internet Engineering Task Force (IETF). *RFC 768. UDP: User Datagram Protocol*, 1980. URL: URL:https://tools.ietf.org/html/rfc768.

[27] Maria Teresa Andrade. Apontamentos da unidade curricular televisão digital e novos serviços, 2015.

[28] Artel. *Broadcasters Guide to SMPTE 2022*. Disponível em: 2016-06-10. URL: https://www.smpte.org/publications/past-issues/January-2015.

[29] The Internet Engineering Task Force. *RTP: A Transport Protocol for Real-Time Applications*, 2003.

[30] Society of Motion Picture e Television Engineers. *IP and Networks. Motion Imagin Journal*, páginas 24–26, Março 2016.

[31] The Internet Engineering Task Force (IETF). *RTP Profile for Audio and Video Conferences with Minimal Control*, 2006. URL: URL:https://tools.ietf.org/html/rfc1890.

[32] The Internet Engineering Task Force (IETF). *RFC4566. SDP: Session Description Protocol*, 2006. URL: URL:https://tools.ietf.org/html/rfc4566.

[33] The Internet Engineering Task Force (IETF). *A General Mechanism for RTP Header Extensions*, 2008. URL: URL:https://tools.ietf.org/html/rfc5285.

[34] Society of Motion Picture e Television Engineers. *SMPTE ST 12-1:2014 -Time and Control Code*, 2014. URL: URL:http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7291029&filter=AND(p_Publication_Number:7291027).

[35] Network Working Group. *A Universally Unique IDentifier (UUID) URN Namespace*, 2005. URL: URL:https://www.ietf.org/rfc/rfc4122.txt.

[36] Mog-Technologies. *MXFSpeedrail*. Disponível em: 2016-06-15. URL: http://www.mog-technologies.com/mxfspeedrail-xpress-solutions/.

[37] Microsoft. *Windows Server 2012 R2*. Disponível em: 2016-06-20. URL: URL:https://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/.

[38] *Blackmagic Design Pty Ltd.* Disponível em: 2016-06-15. URL: https://www.blackmagicdesign.com/.

[39] *Blackmagic Design. Media Express*. Disponível em: 2016-06-15. URL: URL:https://www.blackmagicdesign.com/products/decklink/mediaexpress.

[40] *OpenCV: Open Source Computer Vision*. Disponível em: 2016-06-15. URL: URL:http://opencv.org/.

[41] Jori Liesenborgs. *JRTPLIB*. Disponível em: 2016-06-15. URL: URL:http://research.edm.uhasselt.be/jori/jrtplib/documentation/.

[42] *TR-03 v. SMPTE 2022-6 - what's the score?* Disponível em: 2016-06-14. URL: URL:https://www.linkedin.com/pulse/tr-03-v-smpte-2022-6-whats-score-alan-mercer.

[43] The Internet Engineering Task Force (IETF). *Session Description Protocol (SDP) Source Filters*, 2006. URL: URL:https://www.ietf.org/rfc/rfc4570.txt.

[44] The Internet Engineering Task Force (IETF). *Associating Time-Codes with RTP Streams*, 2009. URL: URL:https://tools.ietf.org/html/rfc5484.