FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Programação Visual para a Definição de um Estúdio Televisivo Baseado na Tecnologia IP

Margarida Xavier Viterbo

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Maria Teresa Magalhães da Silva Pinto de Andrade (Doutoramento)

Programação Visual para a Definição de um Estúdio Televisivo Baseado na Tecnologia IP

Margarida Xavier Viterbo

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Jorge Manuel Gomes Barbosa (Doutoramento)

Vogal Externo: Paula Maria Marques de Moura Gomes Viana (Doutoramento)

Orientador: Maria Teresa Magalhães da Silva Pinto de Andrade (Doutoramento)

---

11 de Julho de 2019

Resumo

Esta dissertação surge no âmbito da mudança de paradigma que tem vindo a ocorrer na indústria de media nas últimas décadas e em diversas áreas. Mais recentemente, a tecnologia *Serial Digital Interface* (SDI) tem vindo a ser substituída pelas redes IP. Neste contexto, este projeto tem como objetivo a especificação e desenvolvimento de uma arquitetura tecnológica que permita a produção de um programa de televisão utilizando, para além de câmaras e microfones profissionais, apenas equipamento COTS[1] (*Commercial Off-the-Shelf*), nomeadamente computadores ligados a uma rede *Ethernet* e utilizando protocolos IP para comunicarem entre si.

Com esta dissertação pretende-se facilitar a transição da indústria televisiva para as tecnologias IP, tentando simplificar a relação entre o negócio televisivo e a engenharia por detrás do mesmo. Desta forma, pretende-se compreender e estudar mecanismos que auxiliem a representação virtual de uma infraestrutura de um estúdio televisivo baseado em IT, onde os componentes de tal infraestrutura estão interligados por redes IP. O objetivo é o de conseguir introduzir este tipo de tecnologias num meio tão tradicional como o da produção profissional de televisão, onde normalmente as tecnologias generalistas não são facilmente aceites e onde não é comum encontrar os conhecimentos adequados. Nesse sentido, foi desenvolvido um protótipo de um sistema *browser-based* que permite programar visualmente um estúdio televisivo, de forma remota e sem necessidade de ligações físicas entre os diferentes componentes de *hardware*. Este protótipo possui um sistema com uma interface gráfica do tipo *drag-and-drop* que permite a um profissional de um estúdio de produção televisiva especificar, através de um diagrama de nós ligados entre si, uma série de operações a realizar num ou mais fluxos de vídeo/áudio. Estes nós representam as transformações SDI para IP, e vice-versa, bem como os *switchers* que possam ser necessários no *workflow* de media. Assim, é possível construir um *workflow* completo para manipular e transferir fluxos de media sem ser necessário conhecimentos técnicos próprios de IT. A interface permite ainda a adição de novos tipos de nós e a respetiva parametrização, conferindo flexibilidade e escalabilidade ao protótipo.

[1] *Produtos Commercial off-the-shelf* (COTS) são soluções universais empacotadas que são depois adaptadas para satisfazer as necessidades do comprador, em vez de serem soluções feitas à medida do cliente.

Abstract

This dissertation emerges from the shift in the paradigm that has been occurring in the last decades in the media industry in many different areas. Most recently, the Serial Digital Interface (SDI) technology is being replaced by IP networks. In this context, this project aims at the specification and development of a technological architecture and corresponding workflow that allow the production of a television program using, in addition to professional cameras and microphones, only COTS[2] (Commercial off-the-shelf) equipment, namely computers connected to an Ethernet network using IP protocols to communicate amongst themselves.

This dissertation intends to facilitate the transition of the television industry to IP technologies, trying to simplify the relationship between the television business and the engineering behind it. Thus, the objective is to understand and study mechanisms that help the representation of an infrastructure of a television studio based on IT, where its components are connected by IP networks. The goal is to introduce these types of technologies in the traditional environment of professional TV production, where usually such generalist technologies are not so easily accepted and where it is not common to find proper knowledge. With this in mind, it was prototyped a browser-based system that allows someone to visually program a TV studio remotely and wirelessly. This consists of a system with a drag-and-drop graphical interface that allows a professional in a television production studio to specify, using diagrams composed of nodes and wires, a series of operations to be performed on one or more video/audio streams. In this way, it is possible to build a complete workflow for the transmission of media streams without requiring technical knowledge in the IT domain. The interface will also allow adding new types of nodes and their parameterization, thus providing flexibility and scalability to the prototype.

[2] Commercial off-the-shelf (COTS) products are packaged solutions which are then adapted to satisfy the needs of the purchasing organization, rather than the commissioning of custom-made solutions.

Agradecimentos

Em primeiro lugar, gostaria de deixar um agradecimento especial à minha orientadora Maria Teresa Andrade, pela ajuda importante em garantir a qualidade do trabalho final. Gostaria também de agradecer à MOG Technologies pelo voto de confiança e apoio, nomeadamente por parte da Ivone Amorim, João Rodrigues e Alexandre Ulisses. Deixo também uma palavra de apreço ao Tiago Costa, investigador do INESCTEC. Finalmente, gostaria de deixar um obrigado muito grande ao meu orientador na MOG Technologies, Pedro Santos, pelo apoio constante e tempo que dedicou comigo ao projeto que fizeram toda a diferença no resultado final.

Margarida Viterbo

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AMWA	Advanced Media Workflow Association
API	Application Programming Interface
BNC	Bayonet Neill–Concelman
CHIC	Cooperative Holistic View on Internet and Content
COTS	Commercial Of The Shelf
DNS	Domain Name System
EBU	European Broadcasting Union
FFMPEG	Fast Forward MPEG
GUI	Graphical User Interface
HD	High Definition
HDCP	High-Bandwidth Digital Content Protection
HDMI	High-Definition Multimedia Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IEFT	Internet Engineering Task Force
INESCTEC	Instituto de Engenharia de Sistemas e Computadores Tecnologia e Ciência
IOT	Internet Of Things
IP	Internet Protocol
IS	International Standard
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
JT-NM	Joint Task Force on Networked Media
MDSN	Multicast Domain Name System
MPEG	Moving Picture Experts Group
NDI	Network Device Interface
NMI	Networked Media Interface
NMOS	Networked Media Open Specifications
PTP	Precision Time Protocol
RDI	Registry and Discovery Instance

RDS	Registration and Discovery System
REST	Representational State Transfer
SD	Service Discovery
SDI	Serial Digital Interface
SDP	Session Description Protocol
SMPTE	Society of Motion Picture and Television Engineers
TTL	Time To Live
TV	Television
UC	Use Case
UDP	User Datagram Protocol
UHD	Ultra-High Definition
UI	User Interface
UX	User Experience
VDFL	Visual Data Flow Language
VPL	Visual Programming Language

Capítulo 1

Introdução

1.1 Enquadramento

Um estúdio televisivo é um ambiente bastante complexo e que envolve a utilização e comunicação entre os mais variados equipamentos, serviços e tecnologias. Um dos principais desafios prende-se com a excessiva complexidade e dificuldade em transformar conceitos de negócio em conceitos técnicos. A necessidade do conhecimento profundo nas duas áreas (da produção televisiva e engenharia) cria uma elevada necessidade na criação de ferramentas e *frameworks* que permitam traduzir conhecimento entre ambos os domínios. Outro grande desafio na área de engenharia tem sido observado na transição da indústria televisiva para as tecnologias IP [1], o que permite produção televisiva remota e sem fios. Assim, tem-se verificado uma transição de operações realizadas por *hardware* para *software*. Ao virtualizar a produção televisiva, pretende-se tirar partido das suas potencialidades, mantendo a qualidade de serviço e a fiabilidade que os produtos de vídeo esperam de um ambiente controlado e profissional.

Esta dissertação enquadra-se no âmbito de uma atividade que integra um projeto de inovação e desenvolvimento no qual a MOG Technologies e a Faculdade de Engenharia da Universidade do Porto participam. O projeto intitula-se CHIC, sendo a atividade mencionada "*Framework* de conceção, *ingest* e transporte de conteúdos de elevada resolução em ambientes IP". Esta atividade surge da observação de uma recente perda de relevância, bastante evidente, dos conteúdos difundidos linearmente em canais televisivos, aparecendo como cada vez mais importantes outras fontes de conteúdos (televisão diferida, *streaming* online, vídeo *on demand*). Esta maior capacidade de escolha por parte do utilizador leva a que cada fornecedor de conteúdos tenha que se destacar de forma a captar a sua quota de mercado. O conceito é o de mudança radical pelo facto de visar a eliminação do uso de SDI na indústria televisiva, o qual

tem sido o método de transferência de dados usado universalmente por todos os intervenientes dessa indústria, ainda que se pretenda que a sua substituição pela tecnologia IP se efetue de forma gradual. Esta transição já tomou lugar no que toca a sistemas baseados em ficheiros, sendo o próximo objetivo fazer a transição de SDI para IP em emissões em direto (*Live* IP) e para grandes volumes de dados. Assim, a realização deste projeto pode ser a chave para a transformação bem-sucedida da produção de conteúdos audiovisuais em UHD *(Ultra High Definition*) e em direto sob redes IP. Em simultâneo permite trabalhar novos mercados e desenvolver novas experiências, criando uma vantagem competitiva substancial na indústria de media.

Para desbloquear o potencial desta abordagem torna-se necessário considerar os diferentes desenvolvimentos e normas que têm vindo a surgir isoladamente e interligá-los. Por um lado, pretende-se conseguir uma aproximação entre a engenharia e a produção e difusão de media. Por outro lado, procura-se acompanhar e contribuir para a transição de SDI para IP. Esta dissertação vem encontrar uma forma de contribuir para estes dois aspetos em simultâneo, simplificando a produção e difusão de conteúdos audiovisuais para os técnicos nas indústrias de *broadcasting* e procurando, em paralelo, uma forma de o fazer recorrendo apenas à tecnologia IP. Tudo isto foi desenvolvido integrando as normas e protocolos desenvolvidos até ao momento que permitem a normalização da integração de protocolos IP na produção audiovisual. Esta abordagem permite criar uma solução inovadora, intuitiva, acessível, leve, de baixo custo e compatível com os mais variados equipamentos e infraestruturas (devido à implementação de tais protocolos).

1.2 Motivação e objetivos

Não existem muitas dúvidas entre os *broadcasters* que a tecnologia IP oferece benefícios enormes para ambientes de produção televisiva. As oportunidades em jogo são claras: largura de banda para tornar produções remotas de alta qualidade uma realidade, a liberdade operacional para reorganizar instalações com base nas necessidades de um projeto e a flexibilidade de incorporar os formatos desejados. Desta forma, prevê-se que muito em breve SDI será trocado pela tecnologia IP graças aos seus benefícios práticos e financeiros. As possibilidades futuras da tecnologia IP na produção televisiva são inúmeras e prometem revolucionar a forma como se lida com emissões em direto, bem como ter um grande impacto noutras áreas de produção. Para a indústria de *broadcast*, os benefícios de usar a tecnologia IP, incluindo maior flexibilidade e *workflows* mais eficientes, estão a tornar-se cada vez mais evidentes.

No entanto, os *broadcasters* continuam a valorizar o SDI pela segurança e confiança, visto ser uma tecnologia bem mais enraizada no mercado. Para muitos, a transição para *workflows* com base em redes IP parece complexa e arriscada. Por isso, esta mudança tem de se manter sensível às práticas atuais a nível de operações, sendo essencial continuar a incorporar aparelhos SDI nos sistemas de produção baseados em IP, bem como normalizar os processos.

O objetivo será construir ferramentas com uma forte componente gráfica e de usabilidade, de forma a simplificar o manuseamento das mesmas por parte dos profissionais de televisão que não possuem conhecimentos tecnológicos próprios de IT ou IP. Espera-se que tais ferramentas possam provar que não é necessário fazer uma escolha entre a familiaridade, previsibilidade e estabilidade do SDI ou as vantagens oferecidas pelo IP, utilizando normas e especificações que garantem compatibilidade e segurança na utilização da tecnologia IP para produção de media. O caminho a seguir terá como objetivo alcançar, num futuro próximo, uma produção e difusão de conteúdos audiovisuais, UHD, e em direto, completamente baseada em redes IP.

Em suma, com esta dissertação pretende-se facilitar a transição da indústria televisiva para as tecnologias IP, tentando simplificar a relação entre o negócio televisivo e a engenharia por detrás do mesmo. A chave para a simbiose entre as duas áreas será a utilização de programação visual para a interface gráfica. Esta forma de programar permite definir fluxos de programas utilizando diagramas com nós e ligações entre os mesmos, tornando a experiência muito mais interativa e intuitiva e permitindo a um técnico (alguém que não tenha conhecimentos de desenvolvimento de *software*) programar um *workflow* de conteúdos audiovisuais de forma independente e muito mais rápida. Além disso, será utilizada uma plataforma *browser-based* para a construção da interface, conseguindo, assim, um produto leve, compatível e de baixo custo. Por fim, no que toca à utilização estrita de redes *ethernet* e protocolos IP para a difusão de media, de forma a garantir a compatibilidade e universalidade da solução, será implementado o protocolo NMOS[3].

Tudo isto faz desta uma solução extremamente ambiciosa e inovadora, com um potencial enorme para revolucionar a indústria televisiva atual, presa a tecnologias tradicionais. Os aspetos inovadores incluem a aplicação de programação visual numa aplicação de produção televisiva, em cenários *time-critical* e *data-driven*; a possibilidade de alguém não especializado construir um estúdio televisivo virtual e personalizado de acordo com as necessidades de cada projeto; o suporte de IP *Live Broadcast* para grandes volumes de dados (UHD); uma solução leve, simples, compatível e de baixo custo que torna a produção televisiva mais eficiente e flexível.

[3] Família de especificações que permitem a compatibilidade de produtos que utilizam a tecnologia IP para produção/transmissão de conteúdos de media (ver capítulo 2).

Para concluir, os objetivos a atingir com a dissertação são os seguintes:

1. Realizar um levantamento, estudo e análise detalhada das principais tecnologias, *frameworks* e protocolos necessários para a produção televisa virtual utilizando redes IP.
2. Escolha e teste de uma plataforma de programação visual que permita o desenvolvimento de um protótipo *browser-based* para a definição de um *workflow* de media.
3. Implementação dos nós necessários para o protótipo na interface gráfica.
4. Desenvolvimento de uma REST API para conectar o *frontend* ao *middleware* e implementar um protótipo que simula produção e transmissão de conteúdos audiovisuais.

1.3 Solução

Um estúdio de produção televisiva é normalmente constituído por um conjunto de diferentes equipamentos com diversas funcionalidades, nomeadamente *ingest*, edição, (re)codificação e continuidade, sendo cada um destes equipamentos caracterizado por uma série de parâmetros. O protótipo desenvolvido no âmbito desta dissertação permite criar uma representação virtual de uma infraestrutura como a descrita, representando cada equipamento por um nó. Cada nó tem a si associada uma descrição das capacidades do equipamento que representa. A aplicação permite selecionar nós, aceder à descrição das suas capacidades, estabelecer ligações entre os mesmos, configurar os parâmetros necessários à sua utilização, guardar e editar *workflows* e iniciar os mesmos. Na prática, permite a um profissional de TV, definir um *workflow* de uma forma virtual e, sem ter que manipular ou aceder diretamente ao equipamento nem ter conhecimentos sobre tecnologias IT, ativar esses equipamentos e iniciar o *workflow* programado.

Por outras palavras, a solução desenvolvida consiste na implementação de um ecossistema de programação visual para componentes de processamento de vídeo sobre redes IP. Isto constitui num sistema com uma interface gráfica do tipo *drag-and-drop* que permite especificar, através de um diagrama de blocos, uma série de operações a realizar num ou mais fluxos de vídeo, bem como adicionar novos tipos de nós e respetivas configurações e guardar os fluxos de dados criados. O resultado final é o desenvolvimento do *frontend* gráfico para a *framework* mencionada anteriormente (*framework* de conceção, *ingest* e transporte de conteúdos de elevada resolução em ambientes IP).

Além disso. foi estudado e implementado o protocolo NMOS e outras tecnologias, como *mDNS*, que permitem a construção de *workflows* de media utilizando apenas redes IP. Este estudo mais aprofundado nesta área permitiu conectar a interface gráfica desenvolvida com o

middleware onde vai ser feita a produção e transmissão das *streams* audiovisuais. A implementação desta camada intermédia foi conseguida graças ao desenvolvimento de uma REST API que faz a comunicação entre o *frontend* e o *middleware*, consistindo num protótipo inicial do que será, futuramente, a camada de *middleware*. Esta camada inclui deteção de NMOS Nodes, obtenção e ativação de serviços e simulação de *streams* de áudio/vídeo correspondentes às desenhadas na interface gráfica.

Para isto, o trabalho realizado teve como foco a especificação *Discovery and Registration* do protocolo NMOS, que permite registar, descobrir e ativar serviços/equipamentos que estejam disponíveis na rede IP de um estúdio de produção, bem como a obtenção das características desses serviços, a sua requisição, parametrização e ativação de uma forma gráfica.

1.4 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos. No capítulo 2, é descrito o estado da arte, as tecnologias relevantes para o projeto e são apresentados trabalhos relacionados. No capítulo 3, é explicado o projeto que deu origem a esta dissertação e a solução que foi desenvolvida. No capítulo 4 é feita uma explicação aprofundada de como foi implementada a solução. No capítulo 5 encontram-se as conclusões finais sobre a solução desenvolvida e o trabalho futuro para dar continuidade ao projeto. Finalmente, no capítulo 6 encontram-se os anexos onde se pode ver um exemplo das estruturas de dados que são os pilares de todo o trabalho.

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

Antes do desenvolvimento da solução, foi feito um levantamento do estado da arte, com o objetivo de adquirir de um conhecimento mais aprofundado das possíveis tecnologias a utilizar no projeto e das abordagens que podem ser adotadas para desenvolver a solução pretendida e alcançar os resultados desejados. Este capítulo apresenta os resultados desse estudo, permitindo que o leitor fique também a conhecer as tecnologias que existem, bem como outros projetos semelhantes e que serviram como ponto de partida para esta solução. Visto que existem duas componentes fortes nesta dissertação – interface gráfica e *middleware* – este capítulo encontra-se dividido em três secções: interface gráfica, media em redes IP e projetos semelhantes. A primeira foca-se em usabilidade e programação visual, incluindo uma apresentação da ferramenta utilizada para GUI. Nesta secção serão explorados os seguintes tópicos: experiência de utilização, programação visual aplicada a fluxo de dados e a ferramenta Node-RED. A segunda secção apresenta as tecnologias e protocolos necessários para trabalhar com difusão de conteúdos audiovisuais utilizando redes IP, nomeadamente o protocolo NMOS (dando ênfase à especificação de *Discovery and Registration*) e o *multicast* DNS. Na última secção são expostos projetos que fazem uso destas tecnologias e cujo intuito é semelhante ao do protótipo desenvolvido, nomeadamente algumas soluções desenvolvidas pela Sony e o projeto *Riedel NMOS Explorer*.

2.2 Interface Gráfica

2.2.1 Experiência do utilizador

Antes de explorar as ferramentas e tecnologias utilizadas para o desenvolvimento da interface gráfica, é útil clarificar dois conceitos muito importantes e frequentemente confundidos. O design da UX (*user experience*) e da UI (*user interface*) são duas coisas muito diferentes. UX refere-se à experiência de utilização do público alvo do produto, ou seja, foca-se em como o produto funciona e como as pessoas interagem com ele. UI está apenas relacionado com o aspeto visual do produto.

Neste projeto o foco será a UX, pois será utilizada uma ferramenta de programação visual (Node-RED) para implementar o protótipo, revolucionando o processo de produção e difusão de media. Contudo, visto que a ferramenta já existe e tem um aspeto visual próprio, muito pouco da UI será alterado. O foco é a usabilidade, não o aspeto visual.

UX design [2] é o processo de criação de produtos que fornecem experiências relevantes para os utilizadores, considerando o porquê do produto, o que é o produto e como se usa. Este processo permite melhorar a satisfação do utilizador com o produto através do melhoramento da usabilidade, acessibilidade e prazer de utilização gerado na interação com o produto. Um bom design da UX garante que o produto é concebido para facilitar as vidas dos utilizadores, levando a um maior volume de vendas e maior retenção de clientes.

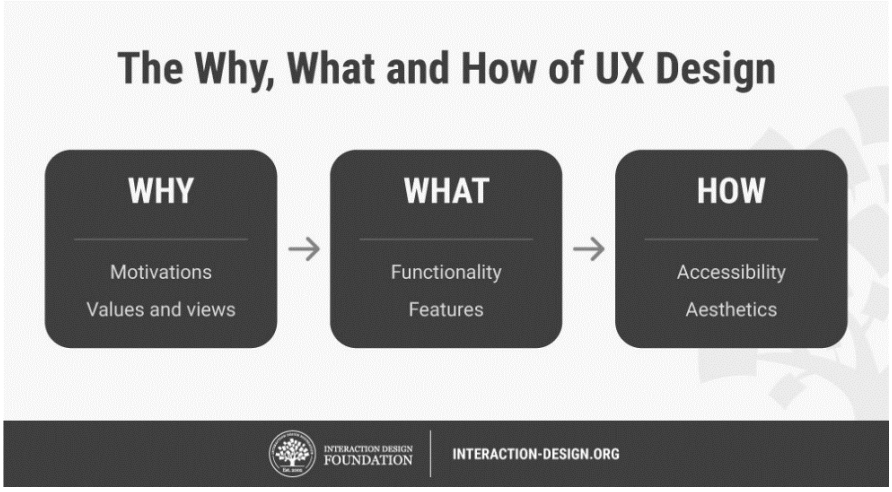

Figura 2.2.1 Os três pilares da experiência de utilização. **[29]**

Os princípios de UX devem estar integrados no desenvolvimento de *software*, de forma a evitar funcionalidades desnecessárias, melhorar a usabilidade do produto, e consequentemente a sua aceitação por parte dos utilizadores, e incorporar objetivos de negócio e *marketing*, protegendo simultaneamente a liberdade de escolha do utilizador.

A experiência de utilização é um tópico muito estudado e trabalhado em desenvolvimento de software para o público em geral. São feitos casos de estudo e ouvidas opiniões do público alvo e de especialistas para garantir que o produto vai agradar às pessoas e ter sucesso. No entanto, quando se trata de ferramentas para um grupo de profissionais mais restrito, software de trabalho para uma área mais específica e onde existe menos concorrência, é frequente a usabilidade receber um pouco menos de atenção. Com a solução desenvolvida nesta dissertação, pretende-se mostrar as vantagens de investir na experiência de utilização para ferramentas profissionais, neste caso para produção e difusão de media. Um software desenvolvido à volta da UX torna o trabalho dos técnicos que o usaram muito mais eficiente, pois conseguem fazer um determinado conjunto de ações de forma mais simples e rápida, diminuindo a duração das tarefas e os erros cometidos. Isto leva a um aumento da produtividade e consequente aumento do rendimento dos trabalhadores e da empresa. A solução desenvolvida é inovadora no âmbito da usabilidade em ferramentas de *broadcast*, pois utiliza a programação visual para montar *workflows* de media, permitindo virtualizar um estúdio de televisão, utilizando um esquema visual e intuitivo. As ferramentas atuais baseiam-se na listagem de aparelhos e painéis de configurações manuais. A abordagem da programação visual contribui significativamente para a diminuição dos tempos e custos de montagem de *workflows* para produção audiovisual.

2.2.2 Programação visual aplicada a programação baseada em fluxo de dados

Em computação, uma linguagem de programação visual *(Visual Programming Language* – VPL) é uma linguagem de programação que permite aos utilizadores criar programas por manipulação de elementos gráficos do programa, em vez de os especificar textualmente. Uma VPL permite programar usando expressões visuais, arranjos espaciais de texto e símbolos gráficos, usados quer como elementos de sintaxe quer como anotações secundárias.

Uma linguagem de programação visual usa imagens para expressar computação. A sintaxe do programa é representada com pelo menos elementos bidimensionais. Os elementos são geralmente caracterizados por cor, dimensão, localização e forma. A sintaxe da programação visual pode ser representada de várias maneiras, sendo a mais comum através de arcos e caixas.

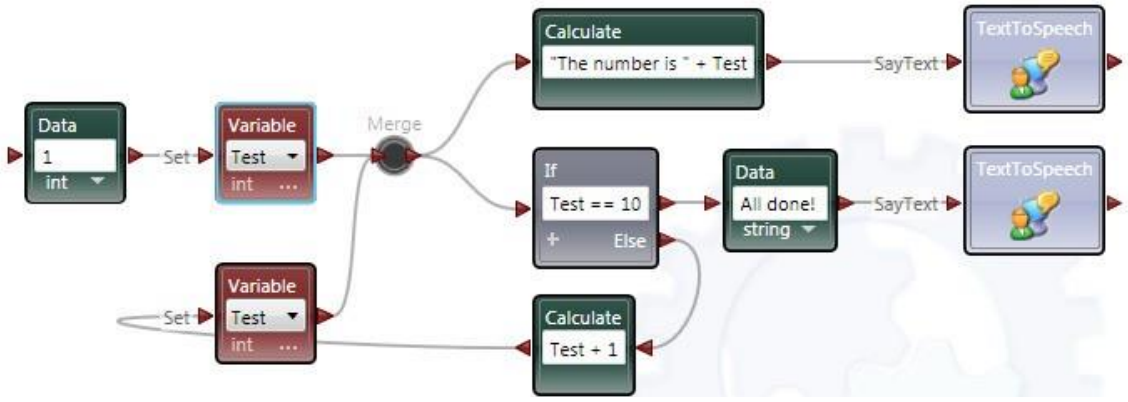

Figura 2.2.2 Exemplo de um programa desenvolvido com a ferramenta *Microsoft Visual Programming Language*. **[33]**

De acordo com a representação de arcos e caixas, um programa visual assemelha-se com um grafo dirigido que consiste em nós computacionais e arcos de dados entre eles. O nó computacional pode ter zero ou mais portas de dados (terminais) e pode ser um nó fantasma, uma unidade operacional ou uma estrutura de controlo. O nó fantasma representa uma interface de entrada ou saída entre o programa e seu ambiente. Unidades operacionais variam desde funções primitivas a estruturas mais complexas que representam elementos (ex. Subprogramas). Um nó de iteração e um nó condicional representam estruturas de controlo que normalmente aparecem num programa visual. Um arco de dados representa um determinado tipo de dados predefinido de acordo com a porta de dados à qual está anexado. Os valores de dados, referidos como *tokens*, fluem através do arco de dados.

As vantagens das linguagens de programação visual residem na representação compreensível de sintaxe, maior facilidade na interação humano-computador, independência linguística, utilização de técnicas de manipulação direta e implementação rápida de software através do uso de prototipagem rápida. As VPLs também têm menos restrições sintáticas e possuem notações visuais que podem ajudar a uma melhor organização, tornando a informação explícita.

Um dos casos de uso mais significativos de uma VPL é a programação baseada em fluxo de dados. Este tipo de programação [3], inventado em 1970 por J. Paul Morrison, é uma forma de descrever o comportamento de uma aplicação como sendo uma rede de nós. Cada nó tem um objetivo bem definido: são lhe introduzidos dados, o nó faz algo com eles e depois envia-os para outro nó. A rede é então responsável pelo fluxo de dados entre os nós. Este modelo encaixa muito bem numa representação visual, tornando-o acessível a um maior número de utilizadores. Se uma pessoa conseguir dividir um problema numa série de passos discretos, consegue olhar para um fluxo e perceber o que este faz sem ter de perceber o código individual de cada nó. Neste contexto surgiram as linguagens de programação visual de fluxo de dados (Visual *Data*

Flow Languages [4] – VDFLs), que permitem paralelização automática e o acesso imediato ao estado do programa, bem como geração automática de código e documentação.

A simbiose entre a linguagem de programação visual e o paradigma do fluxo de dados traz muitas vantagens. A vantagem mais notória é o suporte do paralelismo. Outra vantagem a sublinhar é o rápido ciclo de desenvolvimento de software através de prototipagem rápida, tornado possível pela representação de sintaxe concreta e compreensível. Um grafo direcionado é a maneira mais natural de representar a imagem mental da execução do fluxo de dados. Por exemplo, as dependências de dados entre as instruções são simples e naturais para representar com arcos de dados. Além disso, a possibilidade de inserir facilmente elementos de visualização em pontos diferentes para avaliação dos dados é também bastante útil.

A produção de conteúdo media encaixa-se perfeitamente no paradigma de fluxo de dados, visto tratar-se de um conjunto de aparelhos (nós) a enviar informação entre si através de cabos (fios de ligação). No caso da utilização da tecnologia IP, os cabos físicos não existem, sendo substituídos por ligações em redes *ethernet*. É, portanto, fácil de compreender que a programação visual de fluxo de dados é a abordagem ideal para utilização em ambientes de produção televisiva.

2.2.3 Node-RED

O Node-RED [5] é uma ferramenta de programação visual baseada em fluxo de dados desenvolvida pela equipa de *Emerging Technology Services* da IBM. Foi criado em 2013 e começou por ser uma prova de conceito para visualizar e manipular mapeamentos entre tópicos MQTT[4], tornando-se rapidamente uma ferramenta muito mais geral que pode ser facilmente expandida em várias direções. É um projeto *open source*, fazendo parte da *JS Foundation* desde 2016. Esta ferramenta foi concebida especialmente para ambientes IoT e consiste num sistema *runtime* baseado em Node.js a cujo editor de fluxos pode ser acedido num *web browser*. Aqui é possível criar uma aplicação arrastando nós da palete lateral para o *canvas* e ligando-os. É apenas necessário um clique para fazer *deploy* da aplicação de volta para o ambiente de *runtime* onde o fluxo é executado.

Esta ferramenta foi escolhida para este projeto devido à sua flexibilidade. A palete de nós pode ser facilmente estendida através da instalação de novos nós criados pela comunidade ou

[4] MQTT (*Message Queuing Telemetry Transport*) é um *standard* ISO baseado no protocolo de mensagens *publish-subscribe.* Funciona em cima do protocolo TCP/IP e foi desenhado para conexões em localizações remotas onde é necessária uma pegada de código pequena ou onde a largura de banda da rede é limitada.

desenvolvendo novos nós. Além disso, os fluxos criados podem ser facilmente guardados e partilhados através de ficheiros JSON.

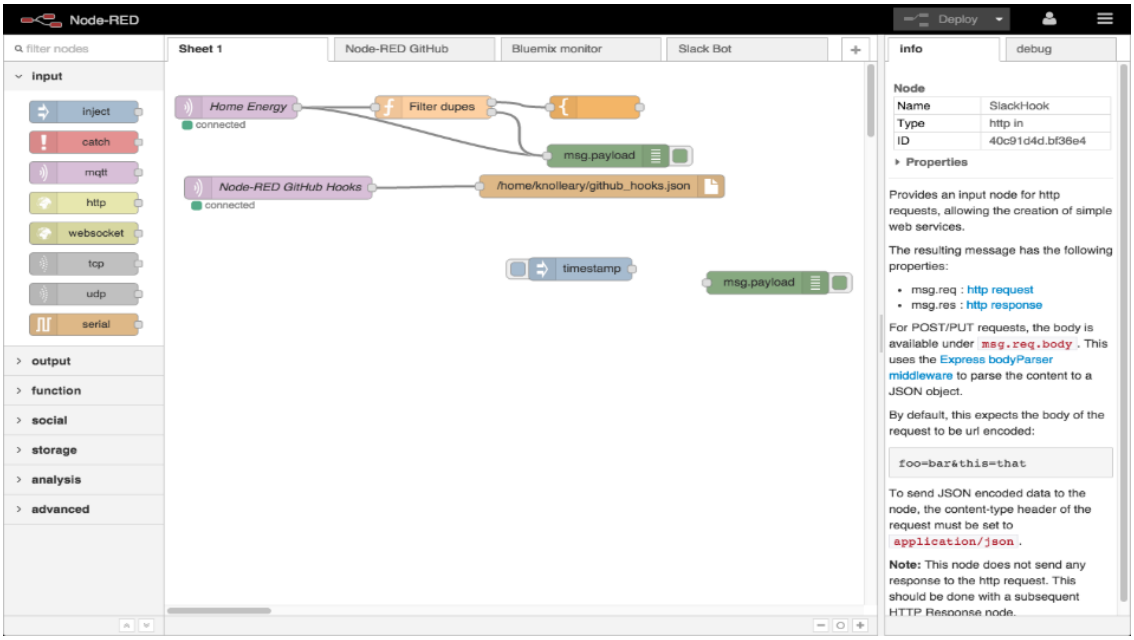

Figura 2.2.3 Interface do Node-RED. **[30]**

O Node-RED fornece um editor *browser-based* que facilita as conexões de aparelhos de *hardware*, APIs e serviços *online*, criando fluxos com uma grande variedade de nós que podem ser implementados em tempo de execução. Esta ferramenta permite também ao utilizador criar e guardar funções dos nós, *templates* e fluxos para mais tarde reutilizar. Isto facilita tremendamente o processo de produção em media. Técnicos da indústria televisiva podem intuitivamente criar *workflows* de trabalho, personalizá-los, guardá-los e reutilizá-los, aumentando significativamente a eficiência de montagem de *worflows* media e *broadcast* de conteúdo televisivo.

Do ponto de vista de desenvolvimento de *software* e integração de aparelhos COTS e tecnologias IP na produção de media, o Node-RED fornece também diversas vantagens. É uma ferramenta leve do ponto de vista do tempo de execução e está construída em Node.js, tirando partido dos seus modelos *event-driven* e sem bloqueios. Isto torna o Node-RED ideal para ser executado integrado numa rede construída em *hardware* de baixo custo. Além disso, os fluxos criados com esta ferramenta são guardados em ficheiros JSON, tornando a sua importação, exportação e partilha muito fácil.

Existem alternativas ao Node-RED, sendo as mais relevantes o *NoFlo* [6] e o *WoTKit* [7].

O *NoFlo* é um ambiente para *Javascript* para desenvolvimento de programas baseados em fluxo de dados, pelo que as instâncias dos seus componentes são representadas por nós. Estes componentes reagem a mensagens sob a forma de *packets* de informação, executando operações predefinidas e enviando respostas no final. Não existe um estado partilhado, a única forma de comunicar é através do envio de *packets*. No *NoFlo* os grafos conseguem lidar com diferentes tipos de paradigmas para os *inputs*: o mesmo fluxo consegue responder a pedidos HTTP, mensagens de texto e alterações no sistema de ficheiros. Da mesma forma, consegue também enviar informação para sítios diferentes, como escrever para uma base de dados, responder a pedidos HTTP ou atualizar a *dashboard*. Existem duas formas para correr os programas com *NoFlo*: se toda a aplicação for baseada em fluxos, utilizar o *NoFlo* para executar e correr os grafos; a outra opção é embutir grafos *NoFlo* na aplicação JS a ser desenvolvida utilizando um módulo NPM.

O *WoTKit* é um conjunto de ferramentas centrado na *web* que ajuda organizações na área de IoT a gerir sensores para colecionar, agregar, guardar e processar os dados recolhidos pelos sensores e reagir a mudanças nos circuitos físicos e virtuais. O *WoTKit* permite publicar, encontrar e usar *streams* de dados de forma simples em visualizações rápidas na *dashboard* do *WoTKit* ou em aplicações desenvolvidas utilizando a *WoTKit* API. A *dashboard* é um ambiente de programação visual baseado em *JavaScript* e permite mostrar uma variedade de visualizações de dados de sensores e componentes de controlo. Para o processamento de serviços é utilizado o *WoTKit Processor*, um subsistema de processamento de dados baseado em eventos. O principal propósito deste sistema é permitir a utilizadores gerarem informação nova e de mais alto nível a partir informação de baixo nível proveniente dos sensores.

Apesar de serem ambas ótimas ferramentas, tanto o *NoFlo* como o *WoTKit* apresentam desvantagens quando comparados com o Node-RED. O *NoFlo* não se encontra direcionado para a edição ou adição de funcionalidades à própria ferramenta como o Node-RED. Para além disso, não é uma ferramenta *browser-based*, mas sim um ambiente de desenvolvimento que permite criar projetos *NoFlo*. Isto leva a problemas de incompatibilidade com outro *software*. Já o *WoTKit*, apesar de também ser *browser-based*, não é tão leve e simples de usar, do ponto de vista do desenvolvimento de *software*, como o Node-RED [8]. Por ser implementado em Node.*js*, o motor de execução do Node-RED corre em *smart devices* e é capaz de executar fluxos com boa *performance* em aparelhos pequenos como uma *Raspberry Pi*.

Por fim, o Node-RED é suportado pela IBM e a sua comunidade de utilizadores é bastante mais significativa do que para as outras ferramentas mencionadas, existindo muito mais documentação, *feedback*, e exemplos de utilização para esta ferramenta. Estas são as principais razões que levaram à escolha do Node-RED para a GUI da solução desenvolvida.

2.3 Media em redes IP

2.3.1 Joint Task Force on Network Media

A *Joint Task Force on Network Media* (JT-NM) foi formada pela *Advanced Media Workflow Association* (AMWA), *European Broadcasting Union*, a *Society of Motion Picture and Television Engineers* e o *Video Service Forum*, no contexto da transição de equipamento e interfaces de *broadcast* construídos à medida para um propósito específico, para redes baseadas na tecnologia IP que servem múltiplos cenários.

A missão da JT-NM [9] é a seguinte: num ambiente aberto e participativo, ajudar a impulsionar o desenvolvimento de uma infraestrutura de rede interoperável baseada em pacotes para a indústria profissional de media, reunindo os fabricantes, emissoras e organizações da indústria (organismos responsáveis por *standards* e normas e associações comerciais) com o objetivo de criar, armazenar, transferir e transmitir media.

A motivação para esta iniciativa baseia-se na crença de que novas oportunidades de negócios serão habilitadas através da difusão profissional de media em redes com base em tecnologias IT a preços acessíveis. A JT-NM foi criada com o objetivo de ajudar a indústria televisiva a fazer a transição para as tecnologias IP, recolhendo os requisitos dos utilizadores, identificando falhas na tecnologia, recomendando boas práticas e coordenando a atividade da indústria. Neste contexto, um dos membros fundadores da JT-NM, a AMWA, criou a *framework* que serviu de chave a esta revolução tecnológica – o protocolo NMOS.

2.3.2 Protocolo NMOS

Uma *framework* que é obrigatório estudar quando se fala da aplicação da tecnologia IP à produção televisiva é o NMOS - *Networked Media Open Specifications*. NMOS é uma família de especificações que estão disponíveis para fornecedores e utilizadores com o objetivo de apoiar o desenvolvimento de produtos e serviços que operam numa *framework* de indústria aberta. Estas especificações foram desenvolvidas pela AMWA e encontram-se disponíveis publicamente no *Github* [10]. A AMWA é uma associação de comércio internacional cujos membros incluem tanto os utilizadores finais como os fornecedores/indústrias, abrangendo desde programadores individuais a empresas multinacionais. Mais recentemente, a AMWA tem alargado a sua atenção para cobrir as camadas de aplicação e controlo para media em redes.

A *framework* NMOS surgiu da observação de que as empresas de media que planeiam comprar soluções de media profissionais baseadas na tecnologia IP enfrentam um leque cada vez maior de soluções não convergentes. Se esta tendência se mantiver, a interoperabilidade

entre produtos de diferentes fornecedores será muito limitada e a capacidade dos utilizadores de construírem melhores sistemas de produção será severamente restringida. O objetivo das *Open Specifications* é fornecer interoperabilidade e aumentar a escolha de produtos entre uma grande variedade de fornecedores, ajudando a indústria dos media profissional a ir ao encontro de um futuro voltado para o *broadcast* baseado exclusivamente nas tecnologias IP.

Desde o início, as indústrias de *live broadcast* têm dependido de tecnologia especializada, como *Serial Digital Interface*[5], e muitos outros protocolos de controlo incompatíveis. Contudo, tem surgido uma mudança no sentido de substituir estas tecnologias por tecnologias IT/IP mais abrangentes, permitindo à indústria beneficiar dos avanços tecnológicos sem comprometer a interoperabilidade de produtos e soluções. Existem algumas especificações para *streaming* de vídeo e áudio através de redes IP. No entanto, nenhuma lida com as camadas de aplicações e controlo, deixando muito trabalho por fazer de modo a ser possível interoperabilidade em ambientes profissionais de media. Isto levou à criação de uma arquitetura de referência para interoperabilidade que, de uma forma básica, identifica modelos e boas práticas a serem aplicados em quatro camadas fundamentais em ambientes media profissionais: Operações, Aplicações, Plataforma e Infraestrutura. A API NMOS [11] é utilizada para fazer a ligação entre as aplicações e a plataforma. Esta API contempla uma vasta lista de especificações para os seguintes campos: descoberta e registo, gestão de conexões entre aparelhos, controlo da rede, eventos e computação, registo de parâmetros, agrupamento, mapeamento de canais áudio e segurança da API.

Nas especificações da NMOS, um *Device* representa um bloco lógico de funcionalidade, enquanto que um *Node* é o anfitrião para um ou mais *Devices*. Os *Devices* têm entradas e saídas lógicas chamadas *Receivers* e *Senders*, sendo todos eles *Resources*. Esta API usa os termos *Flow* para nomear uma sequência de dados (vídeo e/ou áudio) transferidos entre *Nodes* ao longo do tempo. Um *Flow* (composto por *Grains*) liga um *Sender* a um *Receiver* e é tratado como um *Resource*.

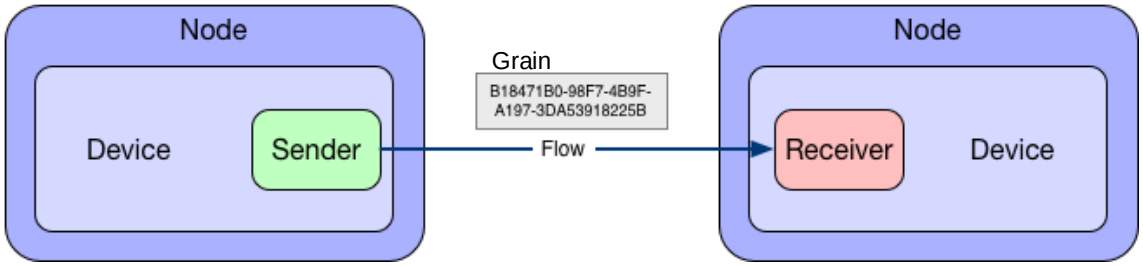

Figura 2.3.1 Esquema ilustrativo dos elementos NMOS. **[31]**

[5] *Serial Digital Interface* (SDI) é uma família de interfaces digitais de vídeo cujos *standards* foram desenvolvidos pela SMPTE. Estes *standards* são utilizados para transmissão de sinais de vídeo não comprimidos e não encriptados dentro de infraestruturas televisivas e todos eles utilizam cabos coaxiais com conectores BNC.

As especificações da NMOS [12] disponíveis publicamente são: *Discovery and Registration Specification* (IS-04); *Device Connection Management Specification* (IS-05); *Network Control Specification* (IS-06); *Event and Tally Specification* (IS-07). De todas estas especificações a mais relevante para este projeto será a de Descoberta e Registo, que servirá de base a todo o projeto.

2.3.2.1 AMWA NMOS Discovery and Registration

Enquanto a indústria televisiva se afasta de um sistema onde tudo é fixo e fisicamente conectado em direção a arquiteturas flexíveis sem fios baseadas na tecnologia IP, torna-se fundamental saber o que está conectado ao sistema e o que pode ser controlado.

Numa infraestrutura de rede dinâmica, a tecnologia IP permite escalabilidade, suporte para múltiplos formatos (atuais e futuros), flexibilidade de localização através de uma infraestrutura distribuída e maior espectro para automação. No entanto, contrariamente ao que acontece em infraestruturas tradicionais, onde origens e destinos são identificados e delimitados por conectores nas matrizes do *router*, numa infraestrutura baseada em IP, surge o desafio da descoberta dos recursos na rede. Assim surgiu a especificação de Descoberta e Registo da NMOS, que garante que as diferentes partes de um sistema de media em rede possam descobrir-se mutuamente. Este é um ponto crucial para automação e redução de tempo de pré-configuração manual, passando de cenários *plug-and-play* físicos para anúncio e identificação automática de serviços numa rede.

Antes das soluções da NMOS, para utilizar a tecnologia IP era necessário utilizar abordagens, como emulação de IP *Routing*[6], que apresentavam diversos problemas: compatibilidade entre aparelhos/*software*, descoberta e controlo dos aparelhos e recursos disponíveis na rede, responsividade dos aparelhos e sincronização de mensagens.

Neste contexto surgiram as especificações NMOS que permitem a interoperabilidade entre aparelhos de diferentes fabricantes, simplificando a integração de aparelhos numa rede: uma implementação funciona com todos os produtos. As principais propriedades desta *framework* passam por: o registo central dinâmico para todos os equipamentos, a identidade do conteúdo é rastreável, a gestão de conexões é unificada e as especificações são completamente públicas, com o objetivo de atingir a maior interoperabilidade possível.

A *framework* NMOS *Discovery and Registration* [13], muito sucintamente, define requisitos/terminologia e descreve a abordagem lógica para:

[6] IP routing é uma metodologia de encaminhamento de *packets* IP dentro e através de redes IP. Isto envolve a determinação de um caminho adequado para um *packet* desde a origem até ao destino numa rede IP.

- Registar informação sobre aparelhos, funcionalidades e recursos
- Anunciar nova informação disponível na rede
- Realizar *queries* sobre essa informação

Em NMOS todos os nós (*logical hosts*) expõem uma ou mais APIs. A *Node* API está presente em todos os nós para permitir encontrar os recursos em cada nó, tendo também um papel fundamental na descoberta *peer-to-peer*. As APIs de Registo e *Query* estão presentes em nós que permitam a descoberta baseada em registo. Os dois requisitos base dos nós são: um nó tem de implementar a *Node* API e tem de tentar interagir com a *Registration* API. Clientes que necessitem de informação sobre os nós no sistema têm de a obter através da *Query* API (se disponível) ou usando as especificações *peer-to-peer* para redes mais pequenas.

Independentemente da implementação, um nó fornece:

- Uma API HTTP para permitir a clientes ver e manipular a informação do nó
- Interfaces através das quais o conteúdo é transportado
- Um PTP *slave*[7] para temporização e sincronização

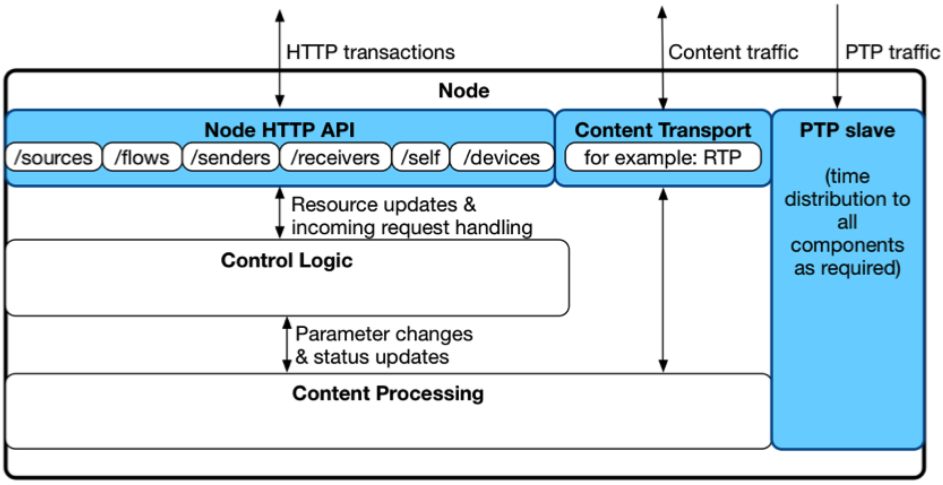

Figura 2.3.2 Representação de um NMOS *Node.* **[31]**

[7] *Precision Time Protocol* (PTP) é a chave para a sincronização de aparelhos de temporização numa rede. Dois componentes essenciais da arquitetura de sincronização hierárquica são o *slave clock* e o *boundary clock*. O primeiro depende do *master clock* para a sua exatidão, enquanto que o segundo pode servir como destino ou origem de informação de sincronização.

A especificação de Descoberta e Registo descreve dois mecanismos para descoberta de nós e respetivos recursos: *peer-to-peer* e nós registados. A descoberta *peer-to-peer* não requer qualquer infraestrutura adicional. Os nós fazem anúncios DNS *Service Discovery* (DNS-SD) relativamente à presença da *Node* API. Os *peers* pesquisam por registos de DNS apropriados e depois preformam um *query* na *Node* HTTP API para obter mais informação. Já o modelo de descoberta por registo acontece usando o *Registration and Discovery System* (RDS), que está desenhado para ser modular e distribuído. Um RDS é composto por um ou mais *Registry and Discovery Instances* (RDIs). Cada RDI inclui o serviço de registo, o serviço de *query* e o *backend* do armazenamento de registos.

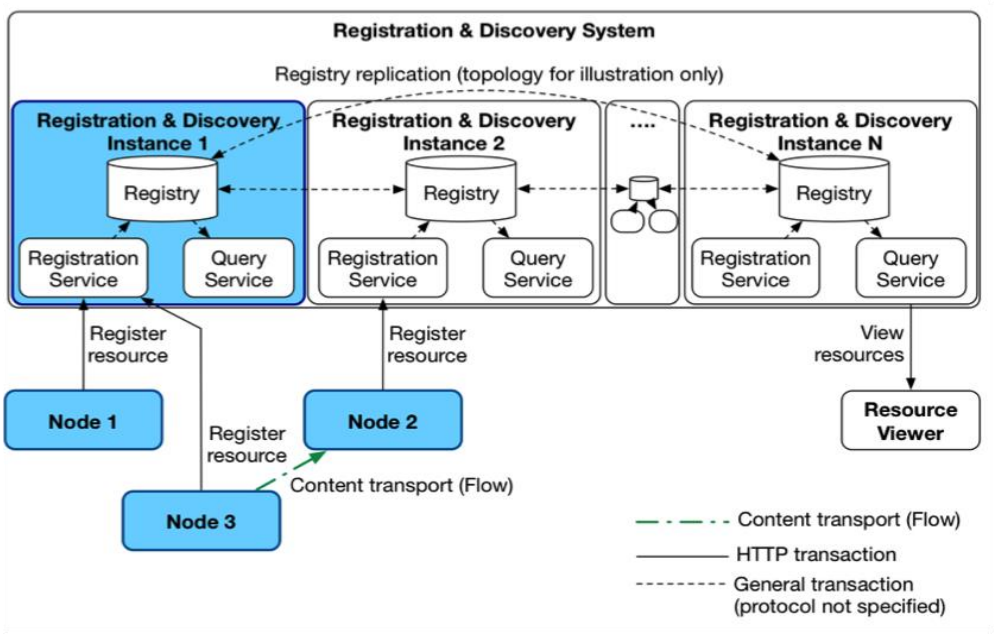

Figura 2.3.3 Esquema representativo do *Registration and Discovery System.* **[32]**

O serviço de registo implementa a API de registo da especificação NMOS de Descoberta e Registo. Os nós fazem um POST para esta API para se registarem a si e aos seus recursos. O serviço de registo também gere os recursos dos nós obrigando-os nós a enviar mensagens *keep-alive* regulares.

O serviço de *query* implementa a API de *query* da especificação NMOS de Descoberta e Registo. Os clientes podem fazer um GET a esta API para obter as listas de recursos. O NMOS *Discovery* utiliza o protocolo DNS *Service Discovery* (DNS-SD) que especifica um mecanismo para o uso de *unicast* DNS ou *multicast* DNS (mDNS) com o propósito de identificar um ou mais *endpoints* numa rede associados com um serviço nomeado.

2.3.2.2 Multicast DNS (mDNS)

Multicast DNS é uma forma de usar interfaces de programação DNS familiares, formatos de *packets* e semântica operacional, numa rede pequena onde não foi instalado nenhum servidor DNS convencional. O mDNS [14] é um esforço conjunto dos grupos IEFT *Zero Configuration Networking* (*Zeroconf*) e DNS *Extensions* (Dnsext). Enquanto que os requisitos para a resolução de nomes da *Zeroconf* poderiam ser satisfeitos criando um protocolo novo, é melhor fornecer essa funcionalidade fazendo o menor número de alterações possível ao *standard* atual do protocolo DNS. Isto salvaguarda os programadores de ter de aprender APIs novas e escrever o código de aplicações de duas formas diferentes - uma para redes de grande dimensão e outra para redes *Zeroconf* pequenas. Isto significa que a grande maioria de aplicações atualmente não precisa de qualquer alteração para funcionar corretamente usando mDNS numa rede *Zeroconf*. Desta forma, os engenheiros não têm de aprender um protocolo novo e as ferramentas atuais de *packets* em redes conseguem descodificar e mostrar *packets* DNS, não tendo de ser alteradas para compreender novos formatos de *packets*.

Em redes de computadores, o protocolo mDNS atribui nomes de anfitriões a endereços IP dentro de redes pequenas que não incluem um *name server* local. É um serviço que não requer configuração, utilizando as mesmas interfaces de programação, formatos de pacotes UDP e semântica operacional do *unicast* DNS.

Quando um cliente mDNS precisa de decifrar um nome de um anfitrião, envia uma mensagem IP *multicast query* que pede ao anfitrião com o nome para se identificar. A máquina alvo faz *multicast* de uma mensagem que inclui o seu endereço IP. Todas as máquinas na sub-rede podem usar a informação para atualizar as suas *caches* mDNS. Qualquer anfitrião pode abdicar da sua reserva do nome de domínio, enviando um *packet* de resposta com um TTL (*time to live*) igual a zero. Uma mensagem mDNS é um *packet multicast* UDP cuja estrutura de *payload* é baseada no formato de *packets unicast* DNS, consistindo em duas partes - o cabeçalho e os dados.

Quando se fala de mDNS, está automaticamente associado à discussão o DNS *Service Discovery* (DNS-SD). A API do DNS_SD faz parte do *Bonjour* [15], uma implementação da *Zeroconf* feita pela Apple. Esta API facilita as seguintes tarefas:

- Registar serviços
- Pesquisar serviços
- Atribuir nomes de serviços a nomes de anfitriões
- Encontrar domínios de serviços
- Atualizar registos

Em suma, mDNS adiciona o serviço de descoberta *multicast* ao DNS (DNS-SD), um processo que consiste na implementação de *Zeroconf* em Node.js. O mDNS fornece uma interface baseada em objetos para anunciar e pesquisar serviços numa rede local. O *multicast* DNS é a chave para o anúncio de equipamentos numa rede e é a base da comunicação entre os serviços na rede IP e o protótipo criado nesta dissertação. Para fazer uso desta tecnologia será utilizado um pacote Node.js que facilita imenso a implementação de *multicast*. Este será utilizado para implementar um pequeno browser para detetar nós NMOS conectados na rede.

2.4 Projetos Relevantes

2.4.1 Soluções comerciais da Sony

Sony IP Live Production

A Sony é pioneira no desenvolvimento de soluções práticas para IP *Live Production* [16] para mercados profissionais. As soluções desenvolvidas pela empresa permitem *streaming* profissional sobre redes IP, *broadcast* de alta qualidade de vídeo e áudio, sincronização de sinais e controlo de dados através da rede IP. Uma das soluções desenvolvida pela Sony - *Networked Media Interface* (NMI) [16] permite transmissão de vídeo baseada em IP para produção live, pós-produção ou produção remota, ligando múltiplas produções ao vivo em diferentes sítios através do IP. O NMI transforma o vídeo em pacotes, áudio e meta data separados, permitindo transmissão em tempo real e alternação limpa entre aparelhos de produção de vídeo através de infraestruturas de rede standard e COTS *switches*.

Os benefícios do NMI incluem:

- Maior flexibilidade e escalabilidade, usando COTS *switches* e infraestrutura IP
- Estrutura redundante de sistemas, permitindo automatizadas de falhas
- Sincronização baseada em redes, eliminando a necessidade de usar *genlocks* separados

A tecnologia da Sony também já chegou a Portugal. Em setembro de 2018 a SIC tornou-se a primeira emissora portuguesa a ter uma infraestrutura para produção de TV e conteúdo multimédia que utiliza apenas a tecnologia IP [17]. A SIC escolheu a Sony para a produção televisiva baseada em IP e distribuição de conteúdo multiplataforma baseada na *cloud*. Esta emissora portuguesa também investiu no sistema de produção de notícias multiplataforma da Sony - *Media Backbone Hive* [18] - permitindo aos seus jornalistas partilhar conteúdo de forma fácil e eficaz.

Sony NMOS C++ Implementation

As soluções acima descritas, embora interessantes no sentido de perceber o que já existe no mercado, não fornecem um bom ponto de partida para esta dissertação pois são produtos comerciais aos quais não é possível ter acesso. No entanto, a Sony tem contribuições *open source* nesta área, nomeadamente uma implementação do protocolo NMOS em C++ disponível no GitHub [19]. Este software está ainda em desenvolvimento, seguindo os progressos contínuos das especificações NMOS, e inclui implementações de um NMOS Node e das API's de Registo, Conexão e *Query*. A aplicação *nmos-cpp-registry* fornece uma instância simples, mas funcional, do NMOS *Registration and Discovery System* (RDS). A aplicação *nmos-cpp-node* fornece um exemplo de um NMOS Node.

Este projeto serviu apenas como auxiliar para o teste do protótipo, tendo sido utilizado para simular aparelhos a correrem nós NMOS, pelo que não será explorado em mais pormenor de forma a não desviar do âmbito do projeto.

2.4.2 Riedel NMOS Explorer

A *Riedel* desenvolveu aquele que será o projeto mais próximo daquilo que foi desenvolvido para esta dissertação. O NMOS Explorer é uma solução não comercial, mas também não *open source*, que apresenta uma interface (ver figura 2.4.1) com uma lista de *Nodes*, *Devices*, *Senders* e *Receivers*. Esta é uma visualização dinâmica que depende da conexão ao *Registry* ou de nós a correr em modo *peer-to-peer*.

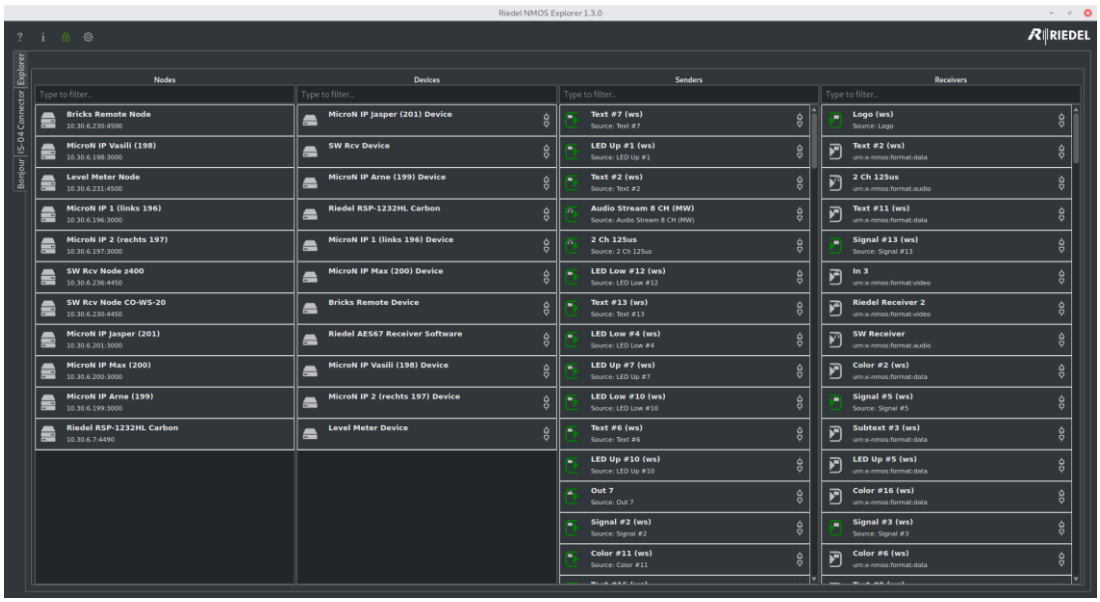

Figura 2.4.1 Interface do *Riedel* NMOS *Explorer*. **[27]**

Esta interface permite selecionar recursos e obter informações sobre o recurso pai ou filho bem como navegar entre os mesmos. É também possível mudar o modo de descoberta, filtrar nós para mostrar apenas nós/aparelhos/*senders*/*receivers* relacionados uns com os outros e ligar/desligar o explorador. Com o explorador ligado, é possível aceder às seguintes funções:

- Carregar num recurso para fazer sobressair os recursos relacionados ou aceder ao *endpoint* da IS-04 Node API correspondente num *web browser*.

- Começar/parar o serviço, ver o respetivo SDP (*Session Description Protocol*)[8], ativar/desativar os seus parâmetros de transporte.

- Criar conexões selecionando um *sender* e um *receiver* e carregando em "*Take*", que vai juntar o *sender* ao target *endpoint* do *receiver*. Todas as conexões são listadas, sendo possível eliminá-las.

A plataforma da *Riedel* é uma ferramenta poderosa e inovadora com todas as funções necessárias para montar um *workflow* em redes IP seguindo o protocolo NMOS. No entanto, apresenta uma grande falha: é pouco intuitiva e torna-se muito complicado montar e guardar *workflows* de forma fácil ou rápida. Este é o grande problema que esta tese vem resolver. Neste projeto serão implementadas funções semelhantes, mas numa plataforma de programação visual. o que facilita imenso a montagem de *workflows* nomeadamente por técnicos de produção televisiva (não-engenheiros).

2.5 Conclusão

Ao longo deste capítulo foram descritas ferramentas e tecnologias que se julgam ser de especial interesse para consolidar conhecimentos estratégicos e identificar abordagens a adotar para a conceção e desenvolvimento da solução desejada. Por um lado, é importante explorar a usabilidade da GUI e a ferramenta Node-RED, que facilita imenso o desenvolvimento da interface. Por outro lado, é também fundamental perceber que desenvolvimentos já existem no tópico da transformação da indústria dos media para as redes IP, quer ao nível de tecnologias (como o protocolo NMOS e o mDNS), quer ao nível de projetos semelhantes ao que foi desenvolvido no âmbito desta dissertação.

Este estudo do estado da arte permitiu selecionar ferramentas e aprender as tecnologias requeridas para este projeto. Em primeiro lugar permitiu perceber o que é e para que serve a programação visual de forma a aplicar os conceitos corretamente na plataforma. Possibilitou também fazer uma escolha informada sobre que ferramenta seria mais apropriada, revelando-se

[8] *Session Description Protocol* (SDP) [28] é um *standard* para definir os parâmetros necessários para a troca de dados audiovisuais (*streaming media*) entre dois ou mais *endpoints*.

o Node-RED a escolha mais inteligente para o desenvolvimento da solução. Por outro lado, este levantamento teórico permitiu obter um contexto para a mudança a correr na indústria de media e as principais entidades a contribuir para uma mudança mais fácil e normalizada, nomeadamente a JT-NM a AMWA.

Assim, percebeu-se de que forma estas organizações estão a contribuir para o problema, estudando os protocolos e APIs que estas definiram. O que levou a um estudo das especificações NMOS de forma a perceber como as utilizar na solução que foi desenvolvida. Aqui o foco foi a especificação de *Discovery ans Registration*, visto ser aquela que trata do problema desta solução: descobrir aparelhos numa rede e transmitir conteúdos audiovisuais entre eles através de redes IP. Para realizar estas operações é necessário utilizar *multicast* DNS, pelo que também foi feita uma breve análise deste protocolo.

Por fim, de modo a perceber de que formas é possível interligar estes conceitos e tecnologias, foram explorados alguns projetos que os utilizam. Descobriu-se uma implementação do protocolo NMOS feita pela Sony (*nmos-cpp*) que ajudou na simulação de aparelhos implementando NMOS numa rede, permitindo fazer um teste do protótipo num ambiente real. Outro projeto interessante que surgiu do trabalho de pesquisa foi o *Riedel NMOS Explorer*. Este projeto, por ter objetivos semelhantes aos pretendidos na solução desenvolvida, foi importante para perceber melhor, utilizando *reverse engineering* (pois o projeto não é *open source*, apenas permite a sua utilização gratuita), como o NMOS pode ser implementado em cenários reais.

Capítulo 3

Exposição do problema

3.1 Introdução e Contextualização

A indústria televisiva está em permanente mudança. Um dos problemas fundamentais é o meio de transporte de sinais. Desde os anos 90 que o SDI tem sido o único método de transporte e tem vindo a desenvolver-se. Mas agora, o transporte nativo em redes IP veio para ficar. Esta tecnologia veio trazer maior escalabilidade, flexibilidade, responsividade e compatibilidade à produção televisiva. No entanto, existe ainda um enorme potencial não aproveitado no uso de redes IP para difusão de conteúdos media. Aliada a uma tecnologia flexível tem de surgir uma forma de a utilizar também flexível, simples e intuitiva, para que produtores televisivos possam tirar maior partido da mesma. É também importante que a produção de conteúdos media possa ser o mais independente possível da engenharia por detrás. Assim surge a necessidade de criar uma interface gráfica capaz de amplificar as vantagens da tecnologia IP e oferecer aos *broadcasters* uma forma de produção e difusão de conteúdos audiovisuais mais simples e eficiente.

Esta dissertação vem oferecer um protótipo de como usar a programação visual para atingir este objetivo. Construiu-se uma ferramenta que, fazendo uso do *multicast* DNS e das especificações NMOS, permite tirar o maior partido da construção de *workflows* de produção e transmissão de media em redes IP. Através de uma interface gráfica de *drag-and-drop* (Node-RED), técnicos de produção televisiva poderão criar e guardar *workflows* muito facilmente e sem necessidade de apoio por parte da engenharia. Esta interface gráfica é atualizada em tempo real com os aparelhos e serviços disponíveis na rede e respetivo estado.

3.1.1 Enquadramento histórico

Os primeiros avanços na indústria audiovisual foram conseguidos com o vídeo analógico. Este foi criado tendo em conta as limitações dos monitores da época: tubos de raios catódicos, sinal preto e branco, com uma taxa de atualização muito lenta e baseados na rede elétrica para sincronização dos sinais. Mais tarde, veio a digitalização, que trouxe uma grande melhoria na robustez, qualidade e transporte de sinais. No entanto, estes ainda eram baseados no formato de onda dos sinais analógicos, pelo que muito do potencial da tecnologia digital ficou desaproveitado. Finalmente chegou o HD e UHD, mas novamente o passo para uma tecnologia mais moderna foi baseado na forma original do sinal analógico, introduzindo uma melhoria apenas em termos de definição. Foram feitas algumas simplificações, obtendo um sinal de vídeo digital, contínuo e linear, mas com uma forma de onde desatualizada. Este é o tipo de sinal utilizado pelo SDI.

Paralelemente ao mundo de comunicações audiovisuais, o mundo das telecomunicações de dados foi também desenvolvido. Os computadores ganharam cada vez mais proeminência em todas as indústrias, a Internet foi criada e com ela, os *standards* para a troca de informação entre sistemas remotos e a tecnologia (nomeadamente largura de banda) foram melhorados, tornando possível o universo digital atual. O detalhe da largura de banda é especialmente importante para esta dissertação, visto que é a maior limitação quando se fala na transmissão de conteúdo audiovisual através de redes de dados.

Surge então a tecnologia IP: um sistema de transporte de media digital que funciona com base nos *standards* de redes IP convencionais. O protocolo IP não será explicado em profundidade pois não faz parte do âmbito desta dissertação. No entanto, há alguns aspetos relevantes a salientar. Uma das características importantes a ter em conta é que o transporte de dados é baseado em *packets*, contrariamente ao sinal de vídeo contínuo usado no SDI, trazendo uma mudança conceptual. Além disso, os *packets* podem ser perdidos ou desordenados, o que implica que tenham de ser implementados métodos que corrijam estes possíveis erros, algo que não era tido em conta antes na indústria televisiva. Estes conceitos são fáceis de entender para um engenheiro informático, mas não tão familiares para um engenheiro audiovisual habituado a lidar com sinais. Por outro lado, existem problemas relacionados com atraso e sincronização que são considerados críticos do ponto de vista audiovisual, mas não têm a mesma importância para um profissional na área de redes e computadores. Poder-se-á então questionar as razões que levam a considerar o IP como um método de transporte de sinais. Este é o tema de discussão na secção seguinte.

3.1.2 Serial Digital Interface versus Internet Protocol

Serial Digital Interface é uma família de interfaces digitais de vídeo que foram normalizadas pela primeira vez em 1989 pela SMPTE (*The Society of Motion Picture and Television Engineers*). Este *standard* tem sido amplamente adotado pela indústria de *broadcast* e é utilizado para a transmissão de sinais de vídeo digitais não comprimidos e não encriptados em estúdios de televisão. À medida que o mundo evolui, o *standard* SDI acompanha a evolução. Novas versões do *standard* têm sido desenvolvidas para dar suporte as melhores resoluções (HD, UHD, 4k, 8k), *bitrates* mais elevadas e mais cores. A última adição à família de interfaces SDI foi lançada em março de 2015, onde foram publicados os *standards* 6G-SDI e 12G-SDI [20], ficando assim a existir oito *standards* SDI. Estes *standards* têm alguns benefícios quando comparados com outros (excluindo o uso de redes IP para *broadcast* de media):

- Comprimento dos cabos: a capacidade de distribuir sinal de vídeo e áudio ao longo de grandes distâncias é um dos benefícios mais proeminentes do SDI. Os seus *standards* usam todos cabos coaxiais com conectores BNC, que permitem transmitir sinais HD/3G-SDI numa distância de 100 metros. Nenhum outro *standard* consegue aproximar-se do SDI neste aspeto.
- Suporte para fibra ótica: optando por fibra ótica como cabo de transmissão, é possível transmitir sinais 4K ou 8K até 10000 metros.
- Conector físico: os conectores BNC utilizados nos cabos coaxiais possuem fechos físicos que oferecem uma durabilidade enorme quando comparados com cabos HDMI (que não os possuem) ou com conectores *DisplayPort* (que apesar de possuírem fechos são muito mais frágeis que os dos conectores BNC).
- Não possui HDCP: *High-Bandwidth Digital Content Protection* é um protocolo de segurança que previne cópia de conteúdo digital (áudio e vídeo) enquanto este atravessa conexões e é utilizado por todos os outros *standards*. Esta proteção requer um *handshake* entre o aparelho fonte e o aparelho de saída, o que resulta em problemas frequentes mesmo em ambientes profissionais. Como o HDCP é bastante lento, o sinal SDI responde mais rápido quando se alterna entre fontes.

Quando comparado com outros protocolos que envolvem ligações físicas entre aparelhos, o SDI tem vantagens, como constatado acima. No entanto, quando comparado com a utilização de redes IP, o SDI apresenta limitações.

No passado, os circuitos de dados tinham capacidade inferior ao tamanho dos sinais de vídeo, por isso o uso de compressão era imperativo. Foi assim que surgiram *standards* como o MPEG-2 e o JPEG2K, que permitiam comprimir o sinal de vídeo original sem perdas, possibilitando a sua reconstrução no fim da transmissão. Desta forma, os sinais podiam ser

transportados num meio com menor capacidade que o sinal reconstruído. Quando a capacidade dos circuitos aumentou, era possível enviar sinais de vídeo não comprimidos através de redes IP. No entanto, isto significa um desperdício da capacidade do meio, não sendo aplicado tudo o que esta tecnologia pode oferecer ao nível de transmissão de dados. Um sinal de vídeo HD transmite muita informação, como sincronização, necessária a ambientes SDI, mas não em ambientes IP. A abordagem de codificar o sinal par IP e depois reconstruí-lo, mantém esta informação desnecessária e gasta largura de banda e recursos. Por essa razão, é preferível a utilização de ambientes de IP nativo, tirando partido das vantagens do uso de *packets* de dados: maior versatilidade, permitindo transporte de resoluções, *standards* e frequências diferentes através do mesmo meio.

Tudo isto, juntamente com a necessidade de ter maior largura de banda para transporte em 4K/8K e o desejo de maior conectividade do que o SDI consegue oferecer levou a que grupos de alianças comerciais e a SMPTE trabalhassem num standard IP designado SMPTE 2110 [21], que veio libertar parte da paralisia verificada na indústria no que toca a decisões de investimento na tecnologia IP. Dois anos antes do SMPTE 2110, a *NewTek* introduziu o NDI (*Network Device Interface*), um *standard* de *software* de vídeo em IP gratuito que se tornou uma opção inicial para as emissoras que desejavam mudar para o IP.

A decisão de adotar o IP ou ficar com o SDI é, em grande parte, influenciada por razões económicas para as emissoras. Para muitos, o protocolo SMPTE 2110, referido anteriormente, pode ser muito caro devido à necessidade de uma infraestrutura de IT totalmente nova, novos produtos IP e recursos humanos adicionais para gerir tudo isto. Apesar de ser um forte substituto para o SDI hoje, muitos dos benefícios do 2110 ainda estão por vir. Assim, algumas emissoras de grande dimensão começam a implementar o padrão SMPTE 2110, mas as emissoras de nível médio e regional podem ter dificuldade em justificar o custo. Este standard estabelece regras para o uso de redes IP para o transporte e produção de sinais audiovisuais, comtemplando problemas como: sincronização, transporte de vídeo não comprimido, transporte de meta data e implementação do protocolo NMOS.

Na tabela seguinte é possível obter um panorama geral das principais diferenças entre os mais relevantes protocolos de produção de vídeo sobre redes IP.

Tabela 1 Comparação entre diferentes protocolos de produção de vídeo em IP. **[22]**

Parameter/Protocol	NDI	NDI HX	NDI HX2	SMPTE 2022-6	SMPTE_2110	ASPEN	NMI
Transport	TCP/UDP/Multi-TCP	UDP (TCP)	UDP, Multi-TCP	UDP	UDP	UDP	UDP
Image Format	Size / Aspect Independent	Size / Aspect Independent	Size / Aspect Independent	SDI only	Size / Aspect Independent		
Tally Feedback	Yes	Yes	Yes	No	No	No	
Compression	NDI Codec (SHQ 2/7)	NDI HX (h.264)	h.264	NONE	(proposed)	NONE	NONE / LLVC Codec
Connection	Socket, Unicast / Multicast and FEC	Unicast/MultiCast	Unicast/MultiCast	MultiCast	MultiCast	MultiCast	MultiCast
HD (1080i) Data Rate	~ 100 Mbit/s	8-20 Mbit/s	~1-50 Mbit/s	>1.5 Gbit/s	>1.1 Gbit/s	>1.5 Gbit/s	>1.5 Gbit/s / up to 14:1[12]
Essence Packing	Discrete Audio, Metadata and Video Frame packets, single connection	Delivered as Discrete Audio, Metadata and Video Frame packets, single connection	Discrete Audio, Metadata and Video Frame packets, single connection	Packetized Raw SDI bitstream	Discrete A, V, Essence and Metadata on separate connections with different protocols	Multiple MPEG Transport Streams	Frame Aligned 2022-6 / LLVC
Infrastructure	Gigabit / Wireless / Load Balanced Multi NIC / 10GBit	Gigabit / Wireless	Gigabit / Wireless	req. 10GBit	req. 10GBit	req. 10GBit	req. 10GBit / Gigabit
Service Discovery	Bonjour (mDNS), NDI Access (manual), Server (NDI4)	Bonjour (mDNS) & NDI Access (manual)	Bonjour (mDNS), NDI Access (manual), Server (NDI4)	NMOS	proposed AMWA IS-04	JSON-RPC	Plug & Play (NDCP)
API	free license, SDK Libraries for Win(x86), Mac, Linux(x86 & ARM), iOS, FPGA reference	hardware encode, Decode with NDI Libraries	Send with NDI Embedded SDK, Receive with Free NDI Libraries	paid-for SMPTE Standards	paid-for SMPTE Standards		paid-for SMPTE RDD

O protocolo SMPTE 2110 ainda se encontra em desenvolvimento, mas não há muitas dúvidas que, no futuro, todo o equipamento usado na indústria televisiva será exclusivamente para redes IP devido às vantagens que esta tecnologia oferece ao simplificar a distribuição de conteúdo de media. Por exemplo, para transmitir vídeo linearmente, ao longo de grandes distâncias, utilizando SDI, existem as seguintes opções: utilizar um circuito de fibra ótica, utilizar ligações de rádio terrestres, utilizar ligações de satélites ou alugar um circuito a um operador. Todas estas opções têm custos elevados e estão muitas vezes indisponíveis em certas localizações. Porém, se forem utilizadas ligações de dados *standard*, redes IP, a transmissão de sinais é possível a partir de virtualmente qualquer sítio e torna-se muito mais barata. Conceitos como produção remota, operação remota, gravação em qualquer localização, edição *on-site* e publicação imediata tornam-se possíveis apenas se aplicados os benefícios que a tecnologia IP oferece. Algumas das vantagens do IP sobre o SDI encontram-se resumidas na tabela seguinte.

Tabela 2 Vantagens da tecnologia IP sobre o SDI.

SDI	IP
Qualquer integração requer uma mudança na infraestrutura.	Fácil migração e integração sem mudanças no sistema.
Necessário hardware pesado.	Apenas requer cabos *Gigabit standard.*
Custos de *hardware* e operacionais elevados.	Excelente relação custo-benefício.
Transmissão de múltiplas *streams* requer investimento em *hardware* e consome tempo.	É possível codificar, transmitir e receber múltiplas *streams* de vídeo e áudio simultaneamente e em tempo real.
Elevado consumo de largura de banda.	Permite a utilização de compressão, diminuindo os requisitos de largura de banda.
Impacto no CPU depende das *streams* de vídeo.	Baixo impacto no CPU.
Mudanças nas especificações, formatos e *standards* requerem uma mudança ou modificação no sistema.	Independente de mudanças nas especificações ou formatos, aberto a *standards* futuros.
Dependente de atualizações, *upgrades* e compatibilidade de hardware dos fabricantes.	Atualização rápida e fácil, baseada em *software.*

A adoção da tecnologia IP na indústria televisiva é, possivelmente, uma das mudanças mais disruptivas na história da mesma. Atualmente, ainda existem alguns aspetos a ponderar antes de fazer a mudança de SDI para IP. Os fabricantes de *hardware* que passaram anos a desenvolver produtos SDI enfrentam uma questão ainda difícil com o IP - redesenhar toda a linha de produtos ou adicionar conversão de e para IP aos seus produtos. A maioria escolheu a

última hipótese, o que significa que não há melhorias de eficiência ou de largura de banda usando o IP com os seus produtos, apenas aumentando o custo para os clientes na maioria dos casos. Os produtos baseados em *software*, por outro lado, são muito mais flexíveis com o IP. Se houver uma versão de *software* IP disponível, tudo poderá existir no computador e nos domínios de rede usando fluxos de IP nativos.

Em suma, a transição tecnológica de SDI para IP tem sido algo polémica. Ao longo dos últimos anos, muita tem sido a discussão em torno da mudança de operações na indústria audiovisual para equipamentos COTS. Contudo, não deveria ser necessário toda esta controvérsia, visto que o IP não é o adversário do SDI, mas simplesmente o seu sucessor. Tal como o sinal analógico deu lugar ao sinal digital, a respeitável tecnologia SDI está a ser ultrapassada por uma forma mais rápida e versátil de produzir e distribuir conteúdo. À medida que são lançadas no mercado novas soluções de qualidade para produção e *broadcast* de vídeo utilizando equipamentos de IT *standard*, as opiniões acerca da sustentabilidade do IP como fundação tecnológica para o futuro têm vindo a melhorar. Com a animosidade em relação ao IP a dissipar, cada vez mais profissionais na indústria de *media* começam a reconhecer que, apesar da tecnologia SDI ter para sempre um lugar importante na história da produção televisiva, o IP traz vantagens com as quais o SDI não consegue competir.

Assim, muitos na indústria de produção de conteúdo media acreditam que o futuro do vídeo é *software*, computadores e redes [23]. Grande parte da nossa interação atual com as principais emissoras gira em torno da migração da produção de vídeo de *hardware* proprietário e salas de equipamentos de vídeo para soluções somente de *software* e centros de dados. Seja uma implementação de uma *cloud* local virtualizada ou uma implementação de *cloud* pública, o resultado é uma abordagem 100% centrada em IT e IP para o futuro. Escalabilidade, flexibilidade, redundância, *backup* e redução geral de custos são razões para o movimento nessa direção, construindo um trunfo para as emissoras que tentam tirar o máximo proveito dos seus negócios. Isso deixa muito pouco espaço para o SDI no futuro a longo prazo da indústria.

3.2 Abordagem ao problema

3.2.1 CHIC: C1 - Framework para conceção, ingest e transporte de conteúdos de elevada resolução em ambientes IP

Neste ambiente de mudança de paradigmas e tecnologias surge o projeto CHIC [24], desenvolvido por um consórcio de empresas, faculdades e organizações do qual a MOG Technologies faz parte. Este projeto tem 3 eixos de atuação:

- A – Este eixo visa desenvolver tecnologias de apoio à produção e disseminação de conteúdos suportados na *cloud*.
- B – As atividades deste eixo estão organizadas em torno de três pilotos dedicados à preservação digital do património cultural e ao desenvolvimento de novas formas de interação em língua portuguesa.
- C – A atividade deste eixo está organizada em 3 pilotos orientados ao apoio à produção de conteúdos de muita alta resolução, 3D e realidade virtual e conteúdos digitais na valorização do livro.

Esta dissertação encontra-se inserida no primeiro piloto do eixo C: C1 - Framework para conceção, ingest e transporte de conteúdos de elevada resolução em ambientes IP. Este piloto visa desenvolver um sistema de ingestão de conteúdos de televisão em muito alta resolução (4K) em tempo real. Estão a ser utilizadas as normas mais recentes para a implementação de sistemas de contribuição utilizando IP, que visam permitir a utilização da Internet na ligação a grandes distâncias entre os sistemas de captação e a régie dos estúdios de realização. Deu-se também prioridade às tecnologias mais avançadas no desenvolvimento da solução, como forma de resolver os problemas associados à grande quantidade de dados a transmitir sem que haja perdas, requisito fundamental da atividade profissional. Esta atividade está a ser desenvolvida em estreita ligação com a atividade de normalização em desenvolvimento na EBU, SMPTE e AMWA, entre outros.

O principal objetivo deste projeto (atividade C1) é ir ao encontro dos desafios decorrentes do uso do SDI na indústria televisiva. Para isso pretende-se especificar e implementar uma arquitetura tecnológica, e correspondente *workflow*, que permita a produção de um programa televisivo utilizando apenas, para além de câmaras e microfones, equipamento COTS (nomeadamente redes *ethernet* e protocolos IP).

Para materializar este objetivo principal, foram estabelecidos um conjunto de objetivos específicos [25]:

1. Definir um fluxo de trabalho adequado que aproveite a flexibilidade da infraestrutura;
2. Projetar uma arquitetura de sistema, incluindo a criação de abstrações reutilizáveis para grandes blocos funcionais mapeados de acordo com as diretrizes definidas pela arquitetura referência JT-NM;

3. Definir os protocolos de transporte de áudio e vídeo com base no trabalho em desenvolvimento no SMPTE e outros fóruns relevantes, para validar a sua aplicabilidade;
4. Definir protocolos para *login*, descoberta e identificação de objetos lógicos na rede, com base no trabalho em desenvolvimento da AMWA NMOS e outros fóruns relevantes, a fim de validar a correção e aplicabilidade do modelo;
5. Implementar um sistema que garanta a interconexão das câmaras e microfones com a rede IP;
6. Implementar *switchers* de vídeo e áudio baseados inteiramente em *software* e *hardware* COTS que permite operações comuns em relação à produção de programas ao vivo;
7. Implementar um sistema de conversão IP para SDI para permitir a integração do estúdio com um ambiente de transmissão tradicional;
8. Implementar um sistema de supervisão, monitoramento e controlo, que permita a gestão e extração de métricas referentes à operação do sistema.

Para atingir estes objetivos foi definido o seguinte *workflow* de teste para a solução pretendida:

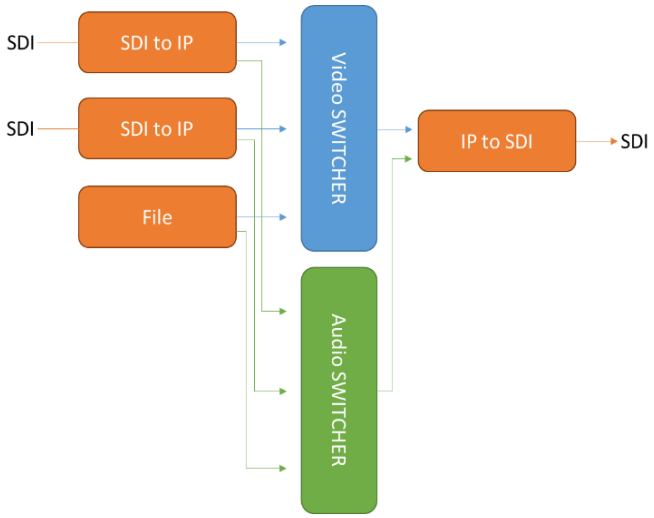

Figura 3.2.1 Esquema representativo do *workflow* de teste da atividade C1. **[25]**

Em termos gerais esta prova de conceito imagina a existência de ficheiros e sinais SDI, que são convertidos para pacotes IP no módulo "SDI to IP". Estes pacotes IP são alimentados a dois blocos separados – o vídeo *switcher* e o áudio *switcher*. Por fim, o processo inverso ocorre com o *multiplexing* dos sinais/ficheiros de volta a SDI no módulo "IP to SDI".

3.2.2 Solução desenvolvida

É no contexto anteriormente explorado que surge a solução aqui apresentada. O protótipo desenvolvido representa a interface gráfica para o projeto descrito na secção anterior. Utilizando uma *framework* de programação visual, permite a alguém fora do ramo de engenharia montar um *workflow* para a produção de conteúdo media, graças à simples e intuitiva forma de utilização das caixas e arcos característicos da programação televisiva. Esta interface gráfica vai fazer a ponte entre a solução desenvolvida (o *middleware* - protocolo NMOS), e o técnico de produção televisiva (o *frontend* desta solução), constituindo a chave para o sucesso da *framework* descrita anteriormente.

Para respeitar os requisitos do projeto em que se insere e ser compatível com o mesmo, esta solução deve seguir algumas regras. Assim, os principais requisitos de sistema a aplicar também a esta solução são os seguintes:

1. Fiabilidade – o sistema deve ser robusto e ter um impacto mínimo nos erros ocorridos durante *broadcasting*;
2. Disponibilidade – o sistema deve estar permanentemente disponível;
3. Modularidade – o sistema deve ser modular;
4. *Upgradability* – o sistema deve possibilitar atualizações e utilizar tecnologias bem verificadas, mas não deprecadas.

A base da solução desenvolvida é a interface gráfica, que foi desenvolvida utilizando o Node-RED. Esta ferramenta permite as funcionalidades importantes de construir um fluxo visualmente de forma simples. Permite também guardar os fluxos criados e carregar para a interface fluxos guardados. Assim, para cumprir os objetivos propostos, foi desenvolvido um pacote Node.js que instala novos nós no Node-RED. Estes nós serão o núcleo do projeto e representam cinco tipos diferentes de aparelhos fundamentais para construir um *workflow* de conteúdo media: *input*/*file distributor*, áudio/vídeo *switcher* e *output distributor*. Os primeiros dois representam *senders* de dados para o fluxo, os *switchers* contêm *senders* e *receivers* para receberem esses mesmos dados trocando endereços IP/portas e reenviando para o *output distributor,* que representa os *receivers* finais do conteúdo media. Em baixo encontra-se uma figura utilizando estes nós e correspondendo ao *workflow* de teste da figura anterior.

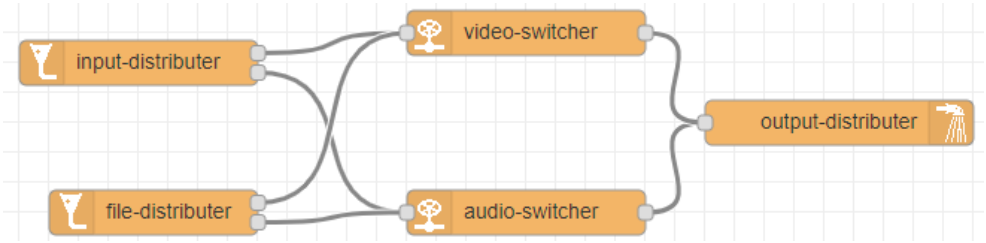

Figura 3.2.2 Fluxo representando uma configuração básica utilizando os cinco módulos desenvolvidos.

Para implementar estas funcionalidades nos nós criados, foi também desenvolvido um servidor em Node.js para fazer a ligação entre o Node-RED (parte visual) e os aparelhos/serviços a correr em nós NMOS. Assim o projeto não consiste apenas na criação visual de *workflows*, tendo sido desenvolvido um protótipo do *middleware* que deverá interagir com a plataforma gráfica.

O protótipo aqui apresentado foi desenvolvido de acordo com as seguintes especificações técnicas:

1. Foco em arquiteturas de referência open standard (ex. JT-NM) que asseguram compatibilidade com outros sistemas;
2. Os endereços IP e as portas devem ser visíveis;
3. Ethernet IP e porta devem ser configuráveis para *streams* media;
4. Agnóstico no que diz respeito à resolução (mesmo que o sistema apenas necessite de suportar uma resolução em cada *runtime*);
5. *Stream switching* em formatos não comprimidos (para áudio e vídeo);
6. Adoção de *multicast* para a transferência de media;
7. Utiliza tecnologia disponível publicamente (não proprietária);
8. Baseado em protocolos *standard*;
9. Desenhado para funcionar com infraestruturas IT atuais.

Capítulo 4

Implementação

4.1 Interface gráfica

A primeira abordagem ao aspeto visual do projeto passou pela instalação e experimentação do Node-RED, para perceber como o seu funcionamento e de que forma poderia ser usado no projeto. O passo seguinte foi criar um pacote Node.js com novos nós que pudessem ser instalados no Node-RED. Finalmente, os nós foram sendo desenvolvidos em paralelo, adicionando as funcionalidades a cada um de forma incremental.

Os nós desenvolvidos neste projeto encontram-se todos juntos por debaixo da secção *mog* na barra lateral esquerda (palete) de nós. Para criar um fluxo basta arrastar os nós para o *canvas* e ligá-los clicando nas portas de entrada/saída de um nó e arrastando até outro. Para guardar o fluxo criado basta clicar em "*Deploy*". Para importar um fluxo, clicar no ícone do menu no canto superior direito, de seguida clicar em "*Import"* a partir do *clipboard* e, finalmente, escolher o ficheiro JSON que contém a informação do fluxo.

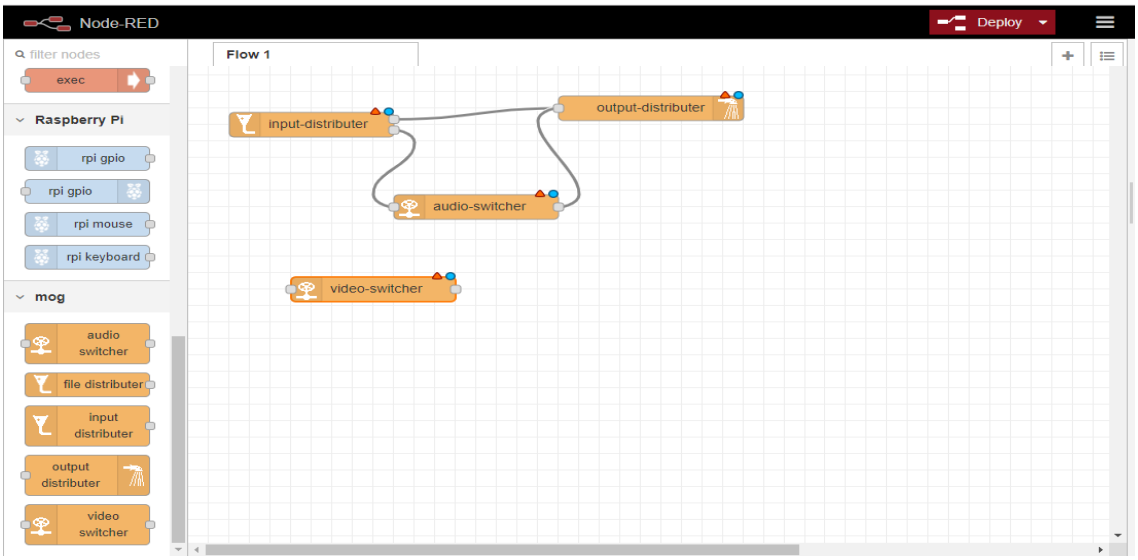

Figura 4.1.1 Interface gráfica da solução.

A parte crucial da interface é o painel de configuração dos nós onde se podem definir os parâmetros necessários para a utilização de *Devices*/*Services* NMOS virtualizados na interface gráfica. O *input-distributor* e o *file-distributor* são muito semelhantes: possuem apenas saídas de vídeo e áudio (assinaladas pelas duas bolinhas e as suas etiquetas). No painel de configuração o utilizador pode selecionar o número de saídas de cada tipo que o nó vai ter. Para cada saída surgem três campos de preenchimento obrigatório que são verificados por uma expressão regular: serviço (*receiver*), endereço IP e porta. Para obter a lista de *senders* ativos é necessário escolher um *device* primeiro.

O *output-distributor* é precisamente o oposto: tem apenas as entradas de áudio e vídeo (ambas sinalizadas na mesma bolinha por impedimento do Node-RED em atribuir mais de um *input* a um nó). Cada entrada é caracterizada pelos mesmos parâmetros do *input-distributor*, a única diferença é que os serviços neste caso são *receivers*. Os *switchers* contêm entradas e saídas, mas de apenas um tipo (áudio para o *audio-switcher* e vídeo para o *video-switcher*). Cada uma das entradas(*receivers*) e saídas(*senders*) são caracterizadas da mesma forma que a mencionada anteriormente, sendo também necessário selecionar o *device* e sendo possível, como nos outros nós, escolher o número de entradas/saídas.

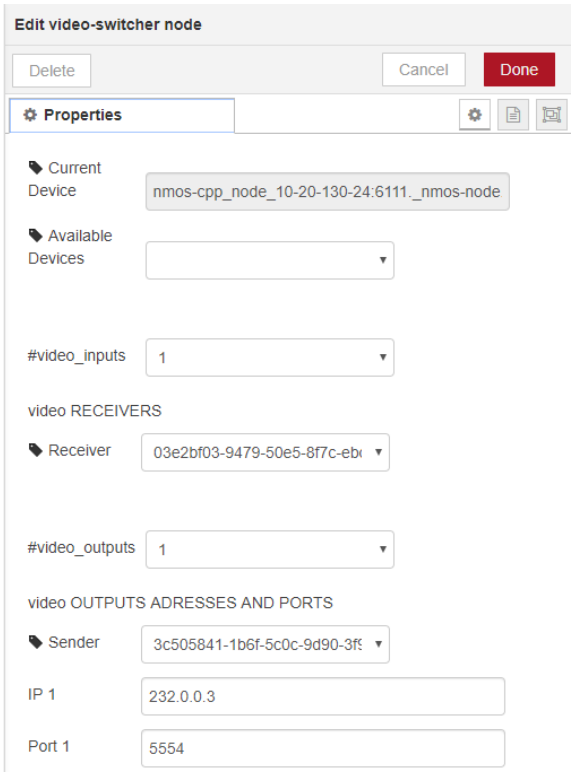

Figura 4.1.2 Painel de configuração para o *video-switcher*.

Os campos de preenchimento de texto dão feedback ao utilizador permanecendo com a borda vermelha até o endereço IP e a porta estarem no formato correto (isto é assegurado por expressões regulares). Ao clicar "*Done*", o painel de configuração fecha e é possível ver uma bola azul no nó indicando que este sofreu alterações. Caso um dos campos esteja incorretamente preenchido, ou nenhum valor tenha sido selecionado nos menus *dropdown*, é assinalado que existem erros na configuração do nó através de um triângulo vermelho. Enquanto existirem nós com o sinal de erro irá, aparecer uma mensagem de erro quando o utilizador clica em "*Deploy*", de forma a alertar para o facto de que o fluxo que está a tentar guardar contém nós mal configurados. Estas pistas visuais são uma ajuda para o utilizador, facilitando a criação de *workflows* corretos, sem ser necessário utilizar métodos empíricos (tentativa erro). As pré-validações de dados permitem também reduzir o número de dados errados que chegam à camada de *middleware*, aumentando a eficiência na construção de fluxos de dados audiovisuais.

4.2 Media em redes

Após um estudo da mudança de paradigma na transmissão de media e do protocolo NMOS, mais especificamente a norma IS-04 *Discovery and Registration*, foram definidos os tipos de nós que seria relevante desenvolver para o protótipo de teste apresentado, bem com as

propriedades que seria necessário definir para cada um. De seguida apresenta-se uma descrição de cada nó:

Input-distributor/File-distributor:
- NMOS *device*
- Número de saídas de áudio
- Número de saídas de vídeo
- Para cada saída de áudio/vídeo:
    - NMOS *sender*
    - Endereço *multicast* IP
    - Número da porta

Output-distributor:
- NMOS *device*
- Para cada entrada de áudio/vídeo:
    - NMOS *receiver*

Audio-switcher:
- NMOS *device*
- Número de entradas de áudio
- Para cada entrada de áudio:
    - NMOS *receiver*
- Número de saídas de áudio
- Para cada saída de áudio:
    - NMOS *sender*
    - Endereço *multicast* IP
    - Número da porta

Video-switcher:
- NMOS *device*
- Número de entradas de vídeo
- Para cada entrada de vídeo:
    - NMOS *receiver*
- Número de saídas de vídeo
- Para cada saída de vídeo:
    - NMOS *sender*
    - Endereço *multicast* IP
    - Número da porta

De forma geral, cada nó vai ter de ter associado a si um NMOS *Device*. Esse *device* contém os serviços (*senders* e *receivers*) que vão ser selecionados para cada entrada/saída. Para que a transmissão de media esteja de acordo com o protocolo NMOS (transmissão sem fios recorrendo a redes IP) será ainda necessário que cada entrada/saída(serviço) de áudio/vídeo esteja associada a um endereço IP (e respetiva porta) de *multicast* para onde *senders* vão enviar dados e os *receivers* vão ler os mesmos). O utilizador apenas necessita de definir os endereços para as saídas, pois, no servidor desenvolvido, vão ser lidos os endereços para as entradas automaticamente (explicado mais à frente).

Para perceber melhor como foi desenvolvida a solução e ser mais fácil compreender os detalhes de implementação descritos abaixo, é importante conhecer a estrutura do projeto. Esta pode ser consultada no esquema a seguir.

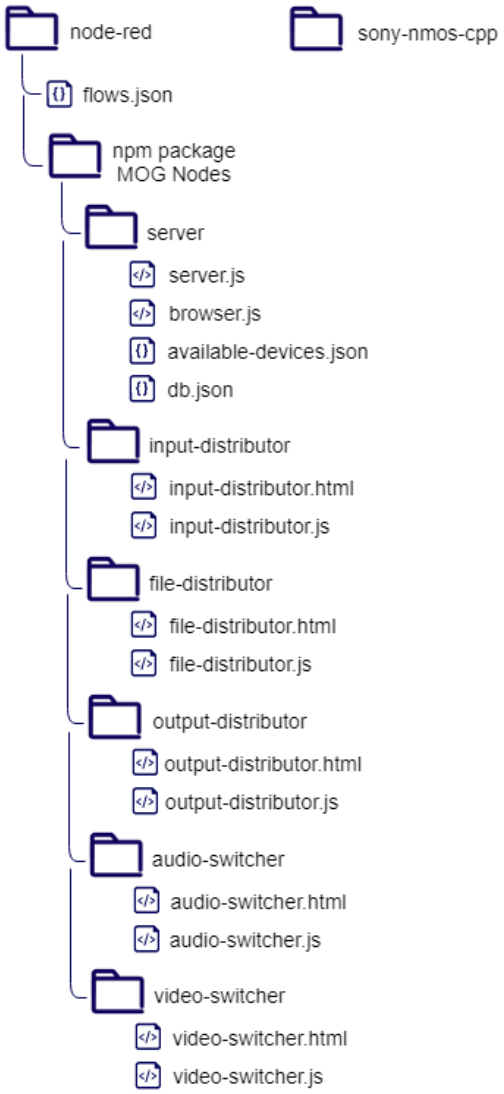

Figura 4.2.1 Estrutura do projeto.

Partindo deste esquema, explicam-se agora os detalhes relevantes de implementação, referindo as pastas e ficheiros acima representados:

- O protótipo desenvolvido é um pacote *Node.js* que inclui a implementação de cada um dos cinco nós mencionados anteriormente e de um servidor (REST API) que serve a interface gráfica, proporcionando a funcionalidade do protótipo e constituindo a ponte entre a GUI e a camada de middleware (NMOS API).

- Como mencionado anteriormente, foi utilizado um projeto *open source* da Sony (*sony-nmos-cpp*) para fazer a simulação de *NMOS Nodes* na rede, de forma a ser possível testar o protótipo.

- Cada nó encontra-se definido por um ficheiro HTML e um *script* em *Javascript*. O ficheiro HTML contém o *frontend* do nó, incluindo parte visual e *triggers* para chamadas à REST API. O *script* em JS contém funções necessárias à construção de um nó para o Node-RED.

- O *backend* (REST API) encontra-se implementado em *Node.js* e é constituído pelo servidor (*server.js*) e um *browser* para deteção de aparelhos NMOS na rede (*browser.js*). Este *browser* utiliza um pacote *npm mdns* para implementar o *multicast* DNS, que permite detetar quando surgem serviços na rede e quando vão a baixo, mantendo uma lista atualizada em tempo real num ficheiro JSON (*available-devices.json* – ver anexo 6.3) .

- O protocolo NMOS está a ser utilizado para detetar aparelhos e serviços, bem como ativá-los. No entanto, o *standard* NMOS aparenta não ter mecanismos de envio de dados para além da ativação dos nós. Assim, para que fosse possível ter um protótipo completamente funcional, foi utilizado o FFMPEG [26] para fazer o envio de dados áudio e vídeo recorrendo ao *multicast* e usando os endereços IP definidos.

- O protótipo é escalável e tem memória persistente. Os dados persistem em memória recorrendo a dois ficheiros JSON: um que o Node-RED gera automaticamente quando se faz *deploy* dos fluxos (*flows.json*), onde é guardado o desenho dos fluxos; outro ficheiro JSON que serve de base de dados alimentada pela REST API (*db.json*), onde são guardadas as propriedades de cada nó. Ambos os ficheiros podem ser consultados em anexo. Os fluxos mantêm-se intactos mesmo após desligar o Node-RED. É também simples acrescentar novos tipos de nós ao projeto ou modificar os existentes. As estruturas de dados permitem

escalabilidade do projeto para quase qualquer tipo de fluxo que se pretenda criar e com um número considerável de nós.

- O servidor e a plataforma GUI são independentes, na medida em que erros ocorridos no servidor não afetam o normal funcionamento da interface, apenas poderão faltar dados que podem ser obtidos repetindo a operação. Os erros são descriminados na consola do servidor, mas este continua operável e capaz de continuar a executar pedidos provenientes da interface. Em suma, podem ocorrer falhas nas operações, mas nem o servidor nem a interface poderão deixar de funcionar abruptamente.

Quando o Node-RED é iniciado são carregados para a interface os nós definidos no ficheiro *flows.json* que está na raiz do Node-RED. A restante implementação pode ser explicada mais facilmente explicando os *triggers* e chamadas à API presentes e as funções que lhes estão associadas. Na figura seguinte, pode ser visto um esquema do funcionamento do protótipo, evidenciando as chamadas à API e os *triggers*.

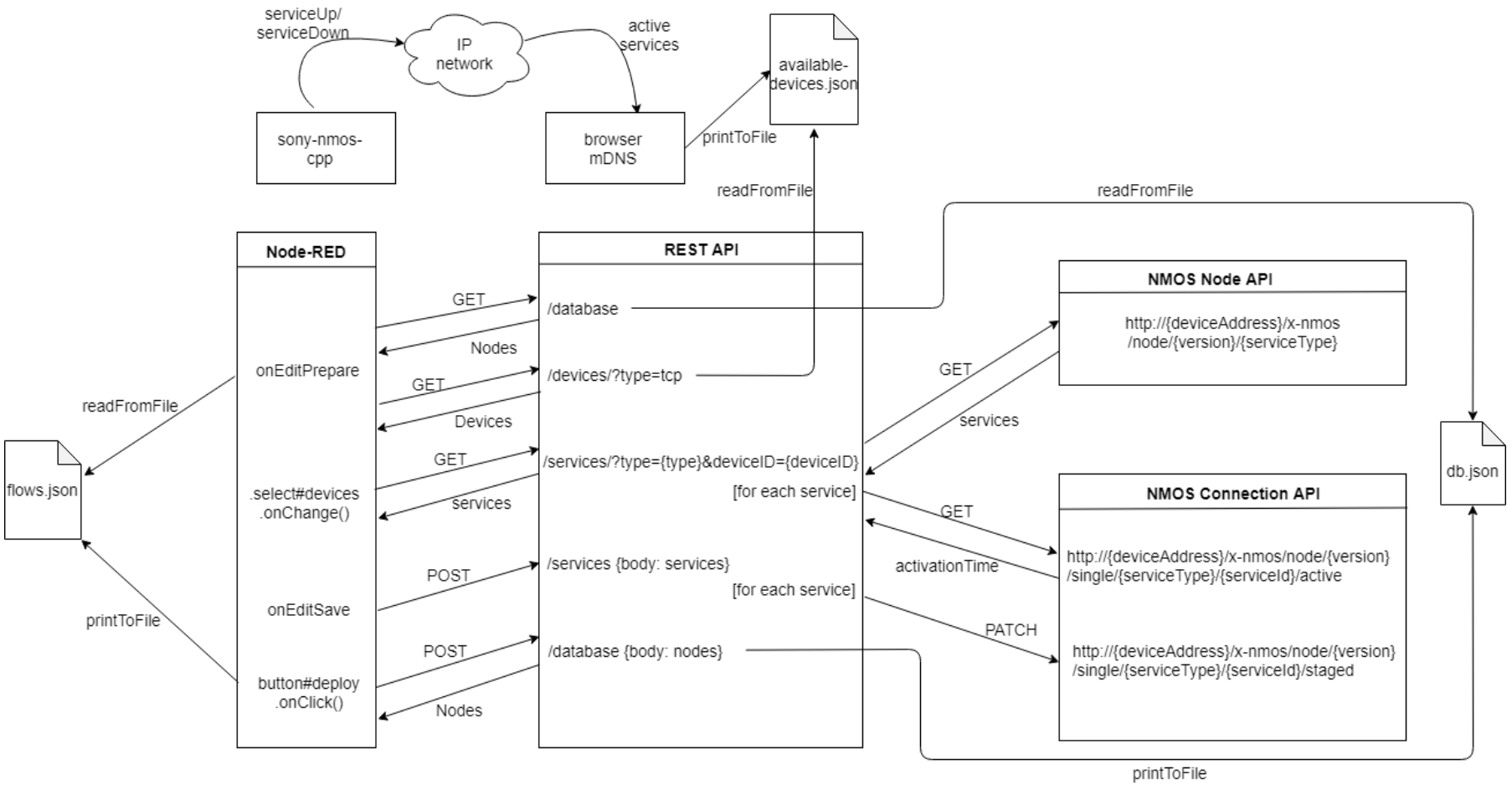

Figura 4.2.2 Esquema representativo do funcionamento do projeto, nomeadamente *triggers* e chamadas à API**.**

Em baixo encontra-se uma explicação mais detalhada do funcionamento do projeto.

- O duplo clique num nó gera um *trigger onEditPrepare* associado ao próprio nó. Este *trigger* serve para preparar a informação que vai ser mostrada no painel de configuração do nó. Se os nós ainda não tiverem sido inicializados localmente, é feita uma chamada à API que devolve a base de dados (*db.json*) e são inicializados os nós. Se o nó que está a ser editado ainda não existir, é adicionado à base de dados. De seguida carrega a informação relativa às entradas/saídas já definidas para aquele nó (se existirem). Finalmente faz uma chamada à API para obter os NMOS *Devices* disponíveis na rede. Por sua vez, este *endpoint* faz uma chamada à NMOS API para obter essa informação e devolver ao nó. A lista de *Devices* vai ser introduzida num menu *dropdown* para o utilizador escolher um.

- O menu *dropdown* para escolha do *device* contém um *trigger onChange* que faz uma chamada à API para obter os serviços disponíveis nesse *Device*. Mais uma vez, a REST API utiliza um *endpoint* da NMOS API para obter os serviços do *device* em questão. Os serviços devolvidos são filtrados de acordo com o tipo (para os *outputs* o pedido à API é para *senders*, para os *inputs* o *request* é para *receivers*) e são distinguidos entre ativos e não ativos, de modo a que depois o utilizador só possa selecionar um serviço não ativo. Assim, o mesmo serviço não pode ser selecionado duas vezes no mesmo nó ou em nós diferentes. A lista de serviços vai então ser usada para preencher os menus *dropdown* de escolha do serviço relativo a cada entrada/saída.

- Carregar no botão *'Done'* gera o *trigger onEditSave* associado ao próprio nó. Este *trigger* serve para guardar todas as propriedades do nó acabadas de definir pelo utilizador, verificando se não existem campos vazios e se os endereços IP e portas estão no formato correto, com a ajuda de expressões regulares. Se algum campo não cumprir os critérios o nó apresenta sinalização de que não está definido corretamente e alerta o utilizador se este tentar iniciar o fluxo (clicar em '*Deploy'*) sem reparar no nó com erros. Após as propriedades gravadas, é feita uma chamada à API enviando a lista de serviços que foram selecionados. Por sua vez a REST API irá, para cada serviço, utilizar um *endpoint* da NMOS API para o ativar.

- Carregar no botão de '*Deploy'* leva ao *trigger* do *eventListener* para o clique, que faz uma chamada à API enviando os nós guardados localmente. A REST API irá atualizar o ficheiro *db.json* com os nós para que estes fiquem guardados em memória persistente. Como referido anteriormente, os endereços IP são apenas definidos para as saídas pois é possível saber quais os endereços das entradas lendo, no ficheiro *flows.json*, os endereços das saídas a que estas estão ligadas. A informação das entradas é completada

nesta fase lendo os nós e ligações no fluxo gravados no ficheiro *flows.json* (automaticamente gerado pelo Node-RED quando se carrega em '*Deploy'*). De seguida os nós atualizados com esta informação são devolvidos à interface que atualiza os seus nós. Isto acrescenta complexidade ao programa, mas torna a interface significativamente mais amigável para o utilizador visto que o utilizador não tem de inserir a mesma informação duas vezes. A outra função importante do '*Deploy'* é criar subprocessos que executam comandos do FFMPEG para a *PowerShell*. Cada nó terá um comando correspondente, começando assim o fluxo de conteúdo media.

Como foi mencionado anteriormente, até ao momento, não parece existir uma forma exequível de enviar informação para nós NMOS para além do pedido de ativação, pelo que não é possível iniciar um fluxo de conteúdo media nestes moldes. Para contornar esta situação, por enquanto, decidiu-se utilizar o FFMPEG para simular o fluxo correspondente ao desenhado na interface do Node-RED. O FFMPEG é uma solução *cross-platform* para gravar, converter e fazer *stream* de áudio e vídeo. Esta *framework* de multimédia consegue descodificar, codificar, transcodificar, filtrar, *mux*, *demux*, *stream* e tocar conteúdo media em qualquer formato. Esta ferramenta foi então usada para produzir áudio e vídeo e transmitir utilizando os endereços *multicast* IP definidos nos nós. Assim, cada nó terá o seu respetivo comando, que será produzido de acordo com o tipo do nó, o número de entradas e saídas de vídeo e de áudio e os endereços *multicast* definidos para cada saída.

4.3 Caso de uso

De forma a explicar um pouco como se utiliza o protótipo e que tipo de soluções podem ser criadas será aqui explicado o caso de uso em que são utilizados um nó de cada tipo como ilustrado na figura 3.2.2.

Neste cenário temos um *input-distributor* e um *file-distributor* a alimentarem o *workflow*. O *input-distributor* representa a transformação de SDI para IP que alimenta o fluxo de media e o *file-distributor* representa a alimentação do fluxo por ficheiros de dados. Temos também um áudio e um vídeo *switcher* para alternar as saídas de áudio e vídeo. Finalmente o conteúdo de media chega ao *output-distributor* que representa a transformação de IP para SDI (ver figura 3.2.1).

Os pré-requisitos para utilizar o protótipo são ter três programas a correr no terminal, não sendo necessário seguir nenhuma ordem em específico. São eles o Node-RED, o servidor (REST API) e o *browser* de serviços NMOS. Como mencionado antes, é também necessário ter nós NMOS a correr, se se pretender testar a utilidade completa do protótipo.

Após estas preparações é muito simples criar um *workflow* digital simulando o *workflow* real constituído por aparelhos. O técnico televisivo terá apenas de arrastar os nós de que necessita (situados no painel lateral) para o *canvas* e ligá-los clicando e arrastando o rato de, e para, os pequenos círculos situados de lado nos nós.

Finalmente será necessário configurar cada nó carregando duas vezes sobre ele e preenchendo corretamente os campos. Para gravar as alterações nas configurações clicar em *'Done'*. Para eliminar nós ou ligações simplesmente clicar cobre os mesmos e carregar na tecla de *delete*. Depois de todos os nós corretamente configurados (os que tiverem erros têm uma pista visual mencionada antes) é só preciso carregar em '*Deploy'* para começar o fluxo de dados.

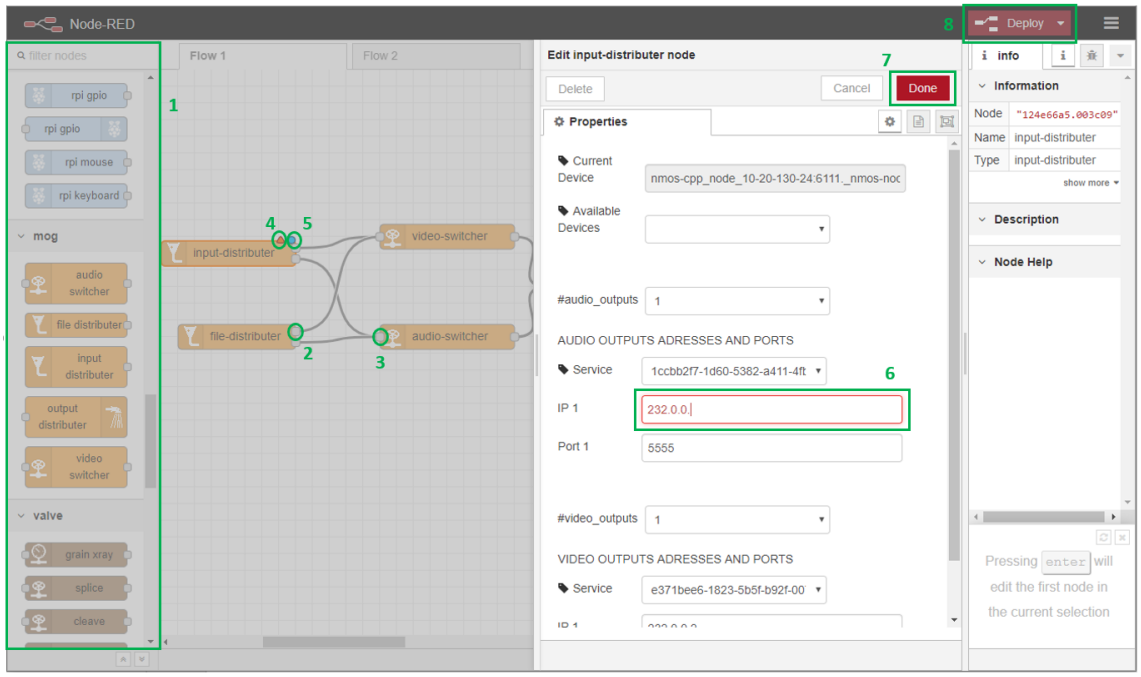

Figura 4.3.1 GUI: 1-Painel de nós; 2-Saída; 3-Entrada; 4-Sinal de erro; 5-Sinal de alteração; 6-Erro no preenchimento do campo; 7-Botão para guardar; 8-Botão de *Deploy*.

4.4 Testes de Validação

De forma a validar todas as funcionalidades da plataforma e cumprimento dos requisitos inicialmente propostos, foi criada uma série de pequenos casos de uso organizados de forma a mostrar o desenvolvimento gradual do exemplo explicado na secção anterior. Assim os testes de validação consistem na verificação do comportamento do protótipo em cada caso de uso (UC).

Cada UC contém mais dois sub-casos de uso que ilustram/validam diferentes aspetos do protótipo. Todos os casos e sub-casos de uso são abaixo descritos, acompanhados de imagens ilustrativas do protótipo. No final desta secção é possível encontrar uma tabela resumo dos UC e respetivo resultado da validação.

Para poder efetuar os testes de validação tiveram de ser previamente iniciados o *browser* de serviços, o servidor e o Node-RED. Para obter serviços na interface foi também iniciado o projeto *sony-nmos-cpp* para simular nós NMOS na rede.

UC1 – Testar adição do input-distributor ao canvas

Testar se é possível adicionar um novo nó ao *canvas* arrastando-o a partir da palete de nós à esquerda para cima do *canvas*.

Figura 4.4.1 Inserir um nó no *canvas*.

UC1.1 – Testar listagem dos senders no input-distributor

Tendo apenas o *input-distributor* no *canvas*, testar se no painel de configurações aparece a lista de *senders* ativos (mas sem estarem em utilização) e permite selecionar um.

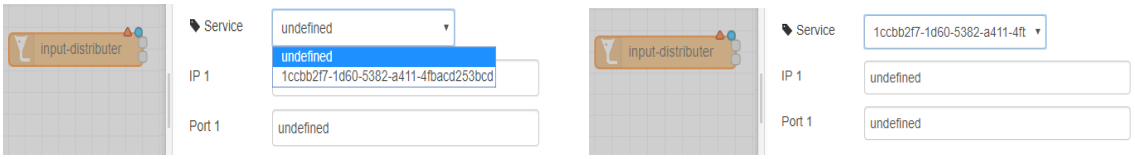

Figura 4.4.2 Listar e escolher *senders*.

UC1.2 – Testar listagem dos Devices no input-distributor

Tendo apenas o *input-distributor* no *canvas*, testar se no painel de configurações aparece a lista de *devices* ativos e permite selecionar um.

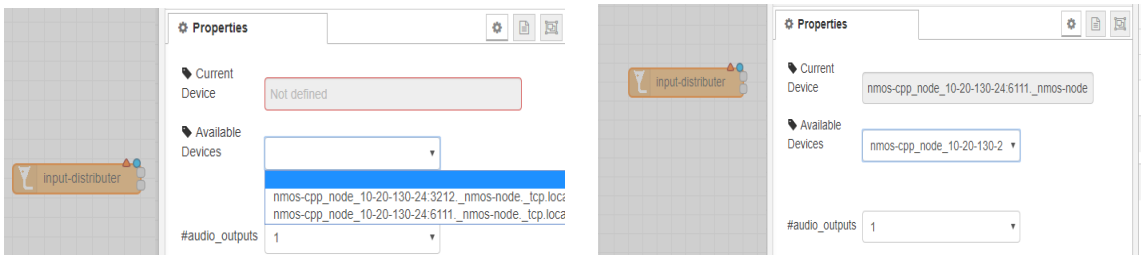

Figura 4.4.3 Listar e selecionar *devices*.

UC2 – Testar inserção de endereço IP e porta no input-distributor

Tendo apenas o *input-distributor* no *canvas*, testar se no painel de configurações é possível definir o endereço IP e a porta de um *output*.

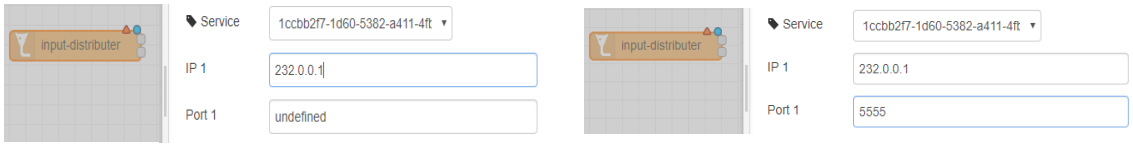

Figura 4.4.4 Inserir endereço IP e porta.

UC2.1 – Testar ligação do input-distributor para o output-distributor

Após a adição do *output-distributor* ao *canvas*, testar se é possível estabelecer as ligações de saída de áudio e vídeo do *input-distributor* para o *output-distributor*.

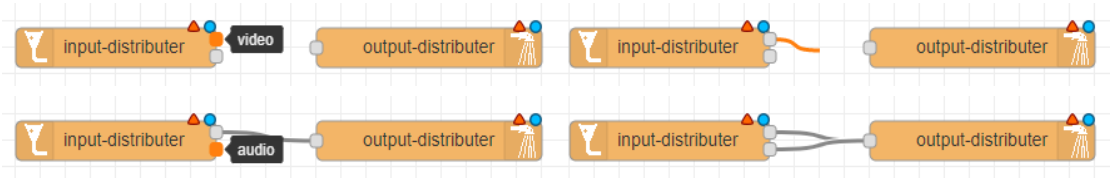

Figura 4.4.5 Estabelecer ligações entre os nós.

UC2.2 – Testar apagar uma ligação do input-distributor para o output-distributor

Tendo o *input-distributor* e *output-distributor* no *canvas*, testar se é possível apagar uma ligação saída do *input-distributor* para o *output-distributor*.

Figura 4.4.6 Apagar ligação entre nós.

UC3 – Testar bloqueio de um serviço que está a ser utilizado por um nó

Após a adição do *file-distributor* ao *canvas*, testar se não é possível selecionar o mesmo *sender* que foi selecionado para o *input-distributor* e que é visível para o utilizador que o *sender* não pode ser selecionado pois aparece a cinzento.

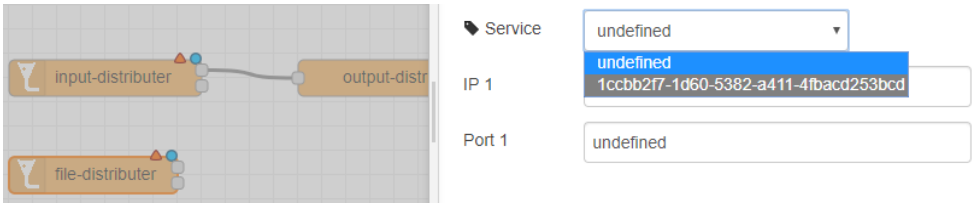

Figura 4.4.7 Bloqueio de serviço em utilização.

UC3.1 – Testar aparecimento de um serviço novo

Testar a adição de um serviço à lista de serviços quando este se torna ativo na rede.

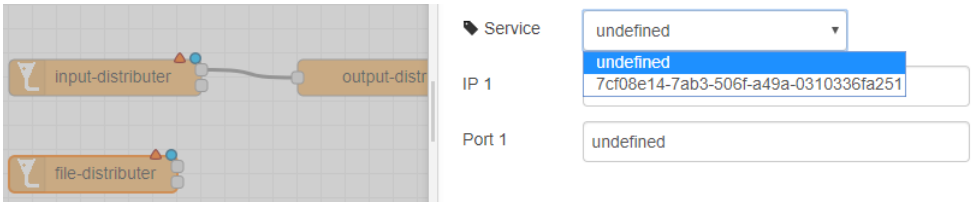

Figura 4.4.8 Aparecimento de novos serviços.

UC3.2 – Testar desaparecimento de um serviço

Testar o desaparecimento de um serviço na lista de serviços quando este deixa de estar ativo na rede.

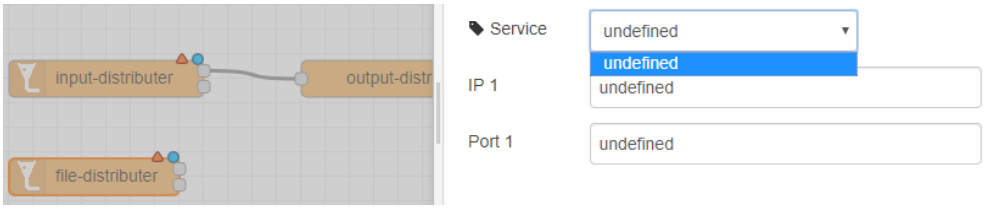

Figura 4.4.9 Desaparecimento de serviços.

UC4 – Testar ligações cruzadas de outputs a inputs acrescentando um audio-switcher

Testar ligações cruzadas fazendo as saídas de áudio passar pelo *audio-switcher* e *output-distributor* receber o áudio a partir do *switcher* e vídeo proveniente de dois nós diferentes.

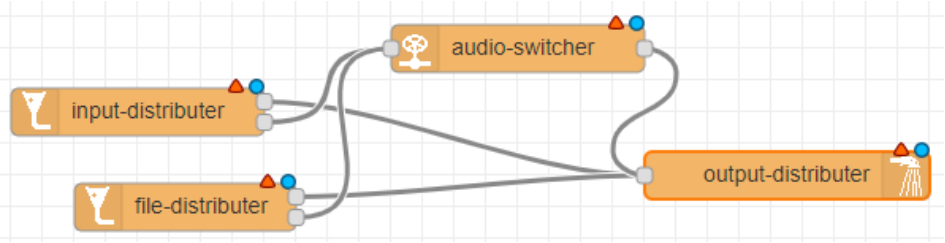

Figura 4.4.10 Múltiplas ligações entre nós.

UC4.1 – Testar se serviços para input(receiver) e output(sender) são filtrados de acordo com o seu tipo

Testar se é feita a diferenciação entre *receivers* e *senders*, mesmo para o mesmo nó quando este tem *inputs* e *outputs*.

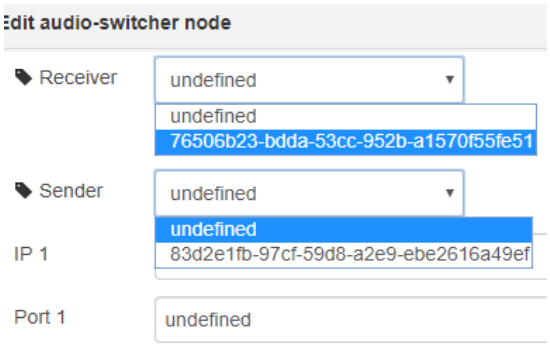

Figura 4.4.11 Seleção de serviços para *input* e *output*.

UC4.2 – Testar se é possível selecionar mais que um input/output para o mesmo nó

Testar se é possível ter mais que um *input* e/ou *output* num nó e que campos necessários à definição de cada entrada/saída surgem na interface.

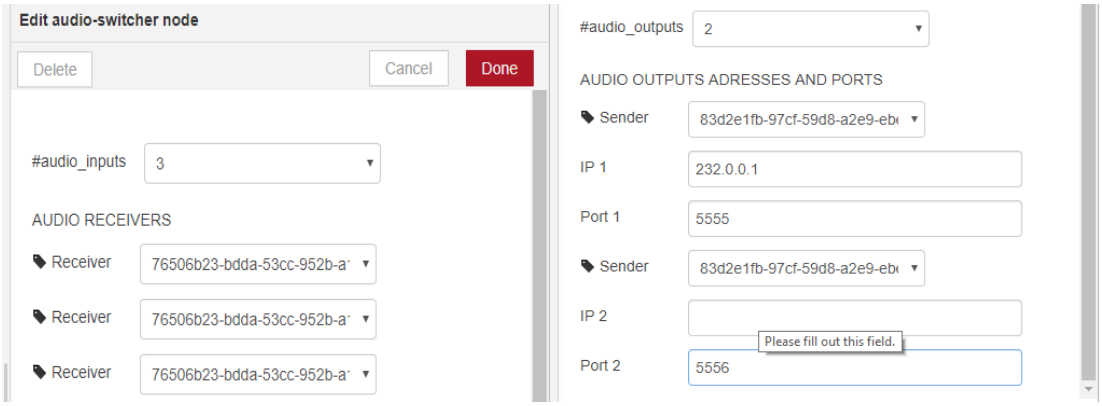

Figura 4.4.12 Selecionar múltiplos *inputs* ou *outputs*.

UC5 – Testar ligações cruzadas de outputs a inputs acrescentando um video-switcher

Testar ligações cruzadas fazendo as saídas de vídeo passar pelo *video-switcher* e *output-distributor* recebe o áudio e o vídeo a partir de cada um dos *switchers*.

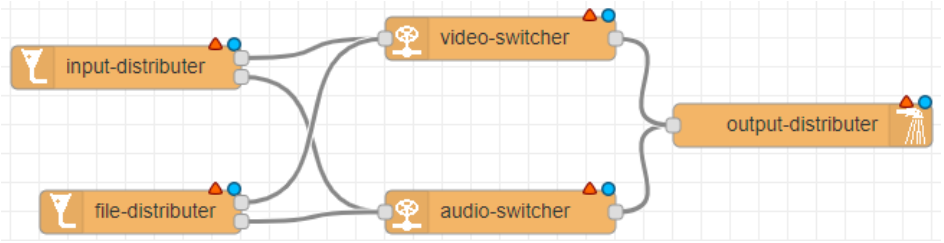

Figura 4.4.13 Ligações entre múltiplos nós.

UC5.1 – Testar validação do input nos campos do endereço IP e porta

Testar se, quando se insere o endereço IP e a porta, os campos mantêm a pista visual da borda vermelha enquanto os valores não estiverem de acordo com o formato correto de endereço e porta.

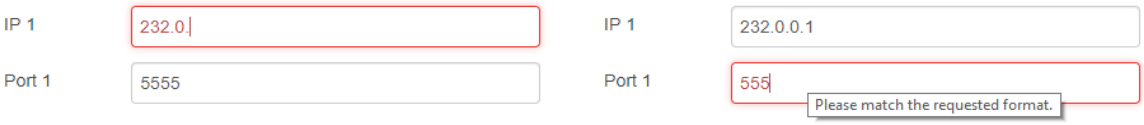

Figura 4.4.14 Validação dos campos de endereço IP e porta.

UC5.2 – Testar gravação de dados ao clicar em 'Done' e respetiva sinalização no nó

Testar se, quando se carrega no botão '*Done*', as propriedades definidas para o nó são gravadas e a sinalização de alteração das propriedades do nó (círculo azul) surge.

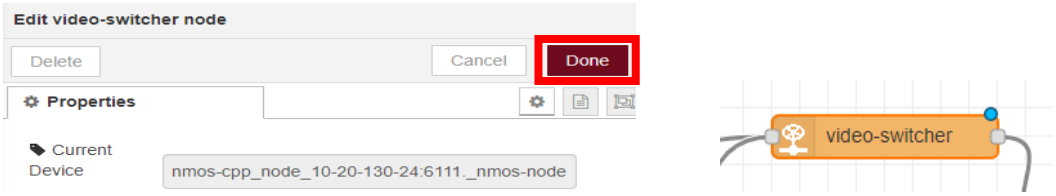

Figura 4.4.15 Gravar alterações e respetiva sinalização.

UC6 – Testar sinalização de erro no nó quando existe uma propriedade mal definida

Testar se o triângulo vermelho que sinaliza um nó mal definido aparece num nó com pelo menos uma propriedade mal definida.

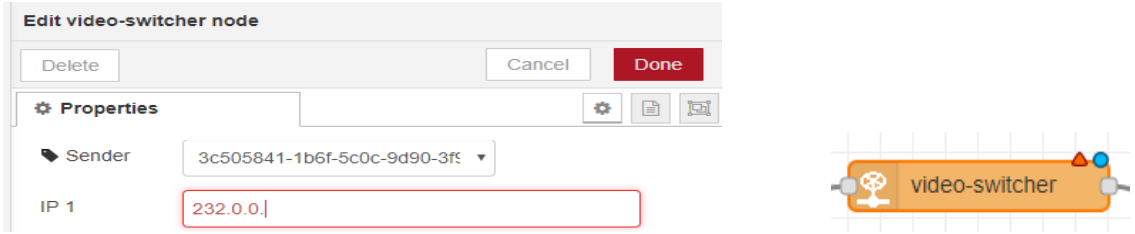

Figura 4.4.16 Sinalização de campo errado e de erro no nó.

UC6.1 – Testar mensagem de erro ao tentar fazer 'Deploy' existindo um nó mal definido no grafo

Testar se, ao carregar no botão '*Deploy*' numa situação em que existe pelo menos um nó com erros, surge um alerta para esse facto.

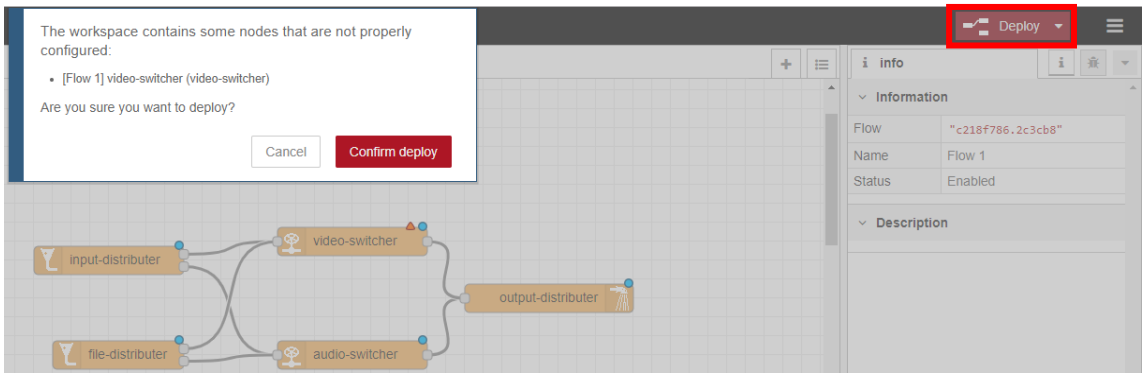

Figura 4.4.17 Mensagem de erro do *Deploy*.

UC6.2 – Testar mensagem de sucesso ao fazer '*Deploy*' não existindo nós mal definidos no grafo

Testar se, ao carregar no botão '*Deploy*' numa situação em que todos os nós se encontram corretamente definidos, surge um alerta para o sucesso da operação.

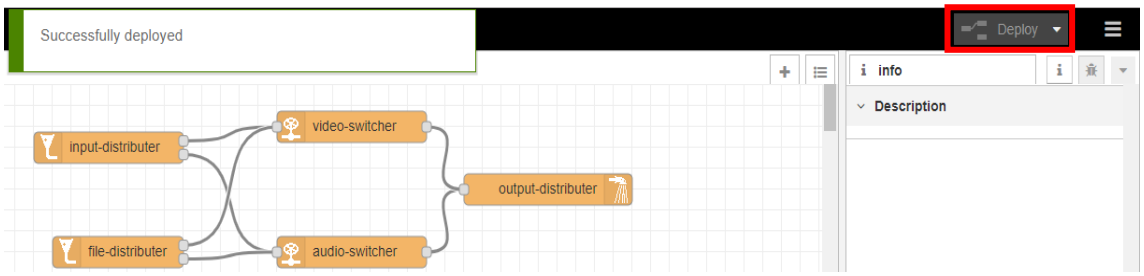

Figura 4.4.18 Mensagem de sucesso do *Deploy*.

O resultado destes testes de validação pode ser consultado na tabela da página seguinte.

Tabela 3 Resumo dos resultados dos testes de validação.

UC	Descrição	Validação
UC1	Testar adição do *input-distributor* ao *canvas*	Passou
UC1.1	Testar listagem dos *senders* no *input-distributor*	Passou
UC1.2	Testar listagem dos *Devices* no *input-distributor*	Passou
UC2	Testar inserção de endereço IP e porta no input-*distributor*	Passou
UC2.1	Testar ligação do *input-distributor* para o *output-distributor*	Passou
UC2.2	Testar apagar uma ligação do *input-distributor* para o *output-distributor*	Passou
UC3	Testar bloqueio de um serviço que está a ser utilizado por um nó	Passou
UC3.1	Testar aparecimento de um serviço novo	Passou
UC3.2	Testar desaparecimento de um serviço	Passou
UC4	Testar ligações cruzadas de outputs a *inputs* acrescentando um *audio-switcher*	Passou
UC4.1	Testar se serviços para *input*(*receiver*) e *output*(*sender*) são filtrados de acordo com o seu tipo	Passou
UC4.2	Testar se é possível selecionar múltiplos inputs/outputs para o mesmo nó	Passou
UC5	Testar ligações cruzadas de outputs a inputs acrescentando um *video-switcher*	Passou
UC5.1	Testar validação do *input* nos campos do endereço IP e porta	Passou
UC5.2	Testar gravação de dados ao clicar em '*Done*' e respetiva sinalização no nó	Passou
UC6	Testar sinalização de erro no nó quando existe uma propriedade mal definida	Passou
UC6.1	Testar mensagem de erro ao tentar fazer '*Deploy*' existindo um nó mal definido no grafo	Passou
UC6.2	Testar mensagem de sucesso ao fazer '*Deploy*' não existindo nós mal definidos no grafo	Passou

Capítulo 5

Conclusões e Trabalho Futuro

Um estúdio de produção televisiva é inerentemente um ambiente distribuído, já que requer equipamento localizado em diferentes pontos devido à natureza das funções a que se destinam, desde aquisição do conteúdo até à continuidade e distribuição, passando pelo *ingest*, transcodificação, recodificação e edição. Assim, a utilização das tecnologias IT e redes IP pode trazer grandes vantagens para a interligação sem fios desses equipamentos e a maximização da eficiência de utilização, sem desvalorizar a diminuição significativa dos custos de produção. Assim, pretendia-se com este projeto criar uma aplicação que permitisse virtualizar a produção de conteúdo televisivo de alta resolução e em direto, usando uma infraestrutura com base no protocolo NMOS e tecnologias IP. Esta aplicação vai permitir que um profissional de um estúdio televisivo possa programar um estúdio televisivo a baixo custo, personalizado de acordo com as necessidades do projeto e de forma autónoma (não sendo necessário um engenheiro), graças à simplicidade e utilização intuitiva que a programação visual oferece.

Relativamente ao desenvolvimento da solução aqui tratada, algumas elações podem ser tiradas. Em primeiro lugar, o Node-RED foi uma ferramenta muitíssimo útil para que o desenvolvimento da GUI fosse um processo muito menos trabalhoso e demorado, pois apenas foi necessário adicionar funcionalidades à interface do Node-RED, em vez de ter de ser desenvolvida uma plataforma de programação visual do zero. No entanto, é possível que existam outras ferramentas semelhantes ao Node-RED que ofereçam mais flexibilidade no que toda à adição ou alteração de funcionalidades. Trabalhar em cima do Node-RED foi o maior desafio no que toca à interface.

No que toca à parte de *middleware*, conclui-se que o NMOS, embora tenha um papel fundamental no desenvolvimento de ferramentas de *broadcast* utilizando redes IP, ainda se encontra um pouco incompleto e/ou difícil de implementar em certos casos. Este problema, juntamente com o facto de as camadas de media e os módulos de *middlewre* do projeto CHIC ainda não estarem

completos, levou à necessidade de utilizar o FFMPEG para validação da interface, construindo um protótipo completo.

Apesar destes obstáculos, o resultado obtido com a solução produzida foi bastante satisfatório e um importante passo para o desenvolvimento de um produto robusto de *broadcast* de media utilizando redes IP. Espera-se que a aplicação revolucione a produção televisiva do ponto de vista financeiro, qualidade de conteúdo e rapidez de produção, oferecendo uma vantagem competitiva considerável a quem dela tirar partido. A virtualização da produção televisiva é o próximo passo necessário na indústria de media. Este projeto pretende juntar-se a outros esforços já em desenvolvimento, tentando aproximar-nos mais de uma realidade acessível de produção plenamente virtual de conteúdo televisivo.

Para além do conhecimento técnico e científico que será gerado, que poderão ser úteis para o setor de media, em particular no domínio de *broadcasting*, a solução desenvolvida contém uma série de características que contribuem para a diferenciar de outras soluções existentes. Os fatores inovadores incluem:

(i) Utilização de programação visual para a construção de fluxos de conteúdos audiovisuais;

(ii) A solução desenvolvida é um sistema aberto (baseado em *open standards*), que conferem interoperabilidade com outras soluções no mercado;

(iii) Permite que uma infraestrutura de IT completa, para *broadcasting* em direto, de conteúdos de alta resolução (elevado volume de dados), seja exclusivamente suportada em protocolos de comunicação IP, com benefícios para os *broadcasters* em termos de redução de custos de operações;

(iv) Permite guardar *workflows* para futuras utilizações;

(v) É uma solução modular, que é possível aplicar a diferentes cenários de utilização;

(vi) Permite a utilização exclusiva de equipamentos COTS, que têm um valor de mercado elevado, numa solução que tem um custo mais acessível;

(vii) Como se trata de uma aplicação *web*, qualquer *browser* é um possível posto e trabalho;

(viii) Suporta validação das configurações dos aparelhos, evitando erros na fase de conceção.

5.1 Satisfação dos Objetivos

Os objetivos definidos para este projeto passavam pela criação de um protótipo de uma interface de programação visual para a criação de um estúdio televisivo virtual. Este protótipo incluiria uma API para comunicação com uma camada de *middleware* que permitiria a difusão de conteúdos de alta resolução em direto utilizando redes IP e implementando os protocolos NMOS. Estes objetivos foram completamente e satisfatoriamente atingidos.

Além disso, foi explorado e desenvolvido ainda mais do que a parte da GUI e respetiva API (proposta inicial da dissertação). Como estava ainda em falta a parte de *middleware* e a ponte para com o mesmo, o escopo da dissertação foi alargado para impulsionar os desenvolvimentos nestas partes e poder criar um protótipo completo e não apenas visual. Isto foi conseguido através da criação da REST API e do *browser* de serviços NMOS.

Desta forma, o trabalho desenvolvido ganhou relevância, ultrapassando o desenvolvimento da parte gráfica para incluir um estudo importante do protocolo NMOS, das suas capacidades e limitações e a incorporação do mesmo no protótipo. Um projeto cujo objetivo seria voltado para a visualização, veio focar-se ainda mais em protocolos IP e no *middleware*, bem como a sua incorporação na GUI desenvolvida, lançando a primeira pedra nesta camada do projeto.

5.2 Trabalho Futuro

Ao nível da GUI muito pouco fica a faltar. Os próximos desenvolvimentos passarão apenas pela criação de mais tipos de nós que venham a ser relevantes ou pequenas alterações nas propriedades dos mesmos, caso os desenvolvimentos na camada do *middleware* a isso o obriguem.

Relativamente aos desenvolvimentos feitos na área de *middleware* e à ponte entre este e a GUI, fica algum trabalho de investigação pendente. Será necessário arranjar uma forma de passar informação aos nós NMOS visto que esta API apresenta muitos *endpoints* para obter informações, mas pouquíssimos para enviar/alterar características dos mesmos. Só após delineada uma solução para este problema poderá ser possível utilizar estritamente e corretamente este protocolo para a construção de *workflows* de media. Nesta altura será necessário fazer alterações à solução aqui projetada, substituído a utilização do FFMPEG pela implementação do fluxo completo do *workflow* recorrendo ao protocolo NMOS.

Anexo A

Exemplos JSON

A1. Estrutura dos nós na base de dados


```json
{
    "nodes": [
        {
            "id": "124e66a5.003c09",
            "type": "input",
            "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
            "n_audio_outputs": 1,
            "n_video_outputs": 1,
            "audio_ips": [
                "232.0.0.1"
            ],
            "video_ips": [
                "232.0.0.2"
            ],
            "audio_ports": [
                "5555"
            ],
            "video_ports": [
                "5556"
            ],
            "audio_services": [
                "1ccbb2f7-1d60-5382-a411-4fbacd253bcd"
            ],
            "video_services": [
                "e371bee6-1823-5b5f-b92f-00722c51f2a3"
            ]
        },
```

```
        {
            "id": "b573fdc7.9a862",
            "type": "input-file",
            "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
            "n_audio_outputs": 1,
            "n_video_outputs": 1,
            "audio_ips": [
                "232.0.0.1"
            ],
            "video_ips": [
                "232.0.0.2"
            ],
            "audio_ports": [
                "5555"
            ],
            "video_ports": [
                "5556"
            ],
            "audio_services": [
                "66aee9b4-d1f8-55de-ae37-830617d8aa04"
            ],
            "video_services": [
                "241ccd2e-2cf3-58be-91ca-e7e779ce607f"
            ]
        },
        {
            "id": "85c01d35.7cc29",
            "type": "audio-switcher",
            "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
            "input_services": [
                "9e3b181d-cf7e-50c1-be0f-db018be83427"
            ],
            "output_services": [
                "83d2e1fb-97cf-59d8-a2e9-ebe2616a49ef"
            ],
            "n_audio_outputs": 1,
            "audio_ips": [
                "232.0.0.4"
            ],
            "audio_ports": [
                "5557"
            ],
            "n_input_audio_outputs": 1,
            "input_audio_ips": [
                "232.0.0.1",
                "232.0.0.1"
            ],
            "input_audio_ports": [
                "5555",
```

```
            "5555"
        ]
    },
    {
        "id": "b55738d5.adaf18",
        "type": "video-switcher",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "input_services": [
            "03e2bf03-9479-50e5-8f7c-ebceb20412bc"
        ],
        "output_services": [
            "3c505841-1b6f-5c0c-9d90-3f9d6d556bfc"
        ],
        "n_video_outputs": 1,
        "video_ips": [
            "232.0.0.3"
        ],
        "video_ports": [
            "5554"
        ],
        "n_input_video_outputs": 1,
        "input_video_ips": [
            "232.0.0.2",
            "232.0.0.2"
        ],
        "input_video_ports": [
            "5556",
            "5556"
        ]
    },
    {
        "id": "17affdaa.67e882",
        "type": "output",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "n_audio_inputs": 1,
        "n_video_inputs": 1,
        "audio_ips": [
            "232.0.0.4"
        ],
        "video_ips": [
            "232.0.0.3"
        ],
        "audio_ports": [
            "5557"
        ],
        "video_ports": [
            "5554"
```

```
            ],
            "audio_services": [
                "bb37a612-3898-59a7-945c-630624187878"
            ],
            "video_services": [
                "24c069f9-ca6b-5505-ae55-c60bb34b5377"
            ]
        }
    ]
}
```

A2. Estrutura dos fluxos no Node-RED


```
[
    {
        "id": "c218f786.2c3cb8",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
    },
    {
        "id": "443a21b6.23fe9",
        "type": "tab",
        "label": "Flow 2",
        "disabled": false,
        "info": ""
    },
    {
        "id": "124e66a5.003c09",
        "type": "input-distributer",
        "z": "c218f786.2c3cb8",
        "name": "input-distributer",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "devices": "",
        "current_device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "validate_outputs": "11",
        "audio_ips": [
            "232.0.0.1"
        ],
        "audio_ports": [
            "5555"
        ],
        "audio_services": [
            "1ccbb2f7-1d60-5382-a411-4fbacd253bcd"
        ],
        "video_ips": [
            "232.0.0.2"
        ],
        "video_ports": [
            "5556"
        ],
        "video_services": [
            "e371bee6-1823-5b5f-b92f-00722c51f2a3"
        ],
        "x": 580,
        "y": 360,
        "wires": [
            [
                "b55738d5.adaf18"
            ],
            [
                "85c01d35.7cc29"
```

```
            ]
        ]
    },
    {
        "id": "17affdaa.67e882",
        "type": "output-distributer",
        "z": "c218f786.2c3cb8",
        "name": "output-distributer",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "devices": "",
        "current_device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "validate_inputs": "11",
        "audio_services": [
            "bb37a612-3898-59a7-945c-630624187878"
        ],
        "video_services": [
            "24c069f9-ca6b-5505-ae55-c60bb34b5377"
        ],
        "x": 1050,
        "y": 400,
        "wires": []
    },
    {
        "id": "b573fdc7.9a862",
        "type": "file-distributer",
        "z": "c218f786.2c3cb8",
        "name": "file-distributer",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "devices": "",
        "current_device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "validate_outputs": "11",
        "audio_ips": [
            "232.0.0.1"
        ],
        "audio_ports": [
            "5555"
        ],
        "audio_services": [
            "66aee9b4-d1f8-55de-ae37-830617d8aa04"
        ],
        "video_ips": [
            "232.0.0.2"
        ],
        "video_ports": [
            "5556"
        ],
        "video_services": [
            "241ccd2e-2cf3-58be-91ca-e7e779ce607f"
        ],
        "x": 590,
```

```
        "y": 460,
        "wires": [
            [
                "b55738d5.adaf18"
            ],
            [
                "85c01d35.7cc29"
            ]
        ]
    },
    {
        "id": "85c01d35.7cc29",
        "type": "audio-switcher",
        "z": "c218f786.2c3cb8",
        "name": "audio-switcher",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "devices": "",
        "input_services": [
            "9e3b181d-cf7e-50c1-be0f-db018be83427"
        ],
        "output_services": [
            "83d2e1fb-97cf-59d8-a2e9-ebe2616a49ef"
        ],
        "current_device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "validate_outputs": "1",
        "validate_inputs": "1",
        "audio_ips": [
            "232.0.0.4"
        ],
        "audio_ports": [
            "5557"
        ],
        "x": 840,
        "y": 460,
        "wires": [
            [
                "17affdaa.67e882"
            ]
        ]
    },
    {
        "id": "b55738d5.adaf18",
        "type": "video-switcher",
        "z": "c218f786.2c3cb8",
        "name": "video-switcher",
        "device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "devices": "",
        "input_services": [
            "03e2bf03-9479-50e5-8f7c-ebceb20412bc"
        ],
```

```
        "output_services": [
            "3c505841-1b6f-5c0c-9d90-3f9d6d556bfc"
        ],
        "current_device": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
        "validate_outputs": "1",
        "validate_inputs": "1",
        "video_ips": [
            "232.0.0.3"
        ],
        "video_ports": [
            "5554"
        ],
        "x": 840,
        "y": 340,
        "wires": [
            [
                "17affdaa.67e882"
            ]
        ]
    }
]
```

A3. Estrutura dos serviços NMOS


```json
{
    "services": [
        {
            "interfaceIndex": 3,
            "typeName": "nmos-node",
            "protocol": "tcp",
            "subtypes": [],
            "fullyQualified": true,
            "replyDomain": "local.",
            "flags": 3,
            "name": "nmos-cpp_node_10-20-130-24:6111",
            "networkInterface": "Ethernet",
            "fullname": "nmos-cpp_node_10-20-130-24:6111._nmos-node._tcp.local.",
            "host": "PC-MOGACADEMY04.local.",
            "port": 6111,
            "addresses": [
                "10.20.130.24",
                "fe80::d05b:b634:8036:1a24"
            ]
        },
        {
            "interfaceIndex": 10,
            "typeName": "nmos-node",
            "protocol": "tcp",
            "subtypes": [],
            "fullyQualified": true,
            "replyDomain": "local.",
            "flags": 2,
            "name": "nmos-cpp_node_10-20-130-24:3212",
            "networkInterface": "Ethernet",
            "fullname": "nmos-cpp_node_10-20-130-24:3212._nmos-node._tcp.local.",
            "host": "PC-MOGACADEMY04.local.",
            "port": 3212,
            "addresses": [
                "10.20.130.24",
                "fe80::d05b:b634:8036:1a24"
            ]
        }
    ]
}
```

Referências

1] M. Poeira, P. Santos, A. Ulisses, D. Costa, P. Ferreira and R. Amor, "An architecture for time-critical IP broadcasting in the cloud".

2] Interaction Design Foundation, "User experience (UX) design," [Online]. Available: https://www.interaction-design.org/literature/topics/ux-design.

3] T. B. Sousa, "Dataflow Programming - Concept, Languages and Applications," 2018.

4] M. Marttila-Kontio, Visual Data Flow Programming Languages: Chalenges and Opportunities, Publications of the University of Eastern Finland, 2011.

5] Node-RED, "Node-RED," [Online]. Available: https://nodered.org/.

6] NoFlo.org, "NoFlo Documentation," [Online]. Available: https://noflojs.org/documentation/.

7] R. L. Michael Blackstock, "WoTKit: A Lightweight Toolkit for the Web of Things," 2012.

8] R. L. Michael Blackstock, "Toward a Distributed Data FLow Platform for the Web of Things (Distributed Node-RED)," 2014.

9] Joint Task Force on Network Media, "Phase two report Reference Architecture v1.0," 2015.

10] AMWA, "Github," 2018. [Online]. Available: https://amwa-tv.github.io/nmos/branches/master/NMOS_Technical_Overview.html.

11] AMWA, "NMOS APIs," [Online]. Available: http://amwa-tv.github.io/nmos-discovery-registration/tags/v1.0/docs/2.0._APIs.html.

12] AMWA, "NMOS Specifications," [Online]. Available: https://github.com/AMWA-TV/nmos/wiki/Specifications.

AMWA, "AMWA NMOS Discovery and Registration Specification," 2016. [Online]. Available:
13] http://amwa-tv.github.io/nmos-discovery-registration/tags/v1.2.1/docs/1.0._Overview.html.

14] "mDns UserGuide," [Online]. Available: http://agnat.github.io/node_mdns/user_guide.html.

Apple, "Bonjour Documentation," [Online]. Available:
15] https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual
/NetServices/Introduction.html#//apple_ref/doc/uid/TP40002445-SW1.

16] Sony, "Technical White Paper: Sony's IP Live Production Technology".

Sony, "Sony Press," 2018. [Online]. Available: https://pro.sony/de_BE/press/sic-selects-ip-based-
17] tv-production-centre-cloud-based-multiplatform-delivery.

Sony, "Media Backbone Hive Reach More Audiences," 2018. [Online]. Available:
18] https://assets.pro.sony.eu/Web/commons/solutions/pdfs/Hive_Guide_02_2018_LR.pdf.

19] Sony, "nmos-cpp," [Online]. Available: https://github.com/sony/nmos-cpp.

Steel Door Institute, "Compilation of SDI Technical Documents and ANSI/SDI Standards and
20] Test Methods Steel Door Institute".

21] TM Broadcast, [Online]. Available: http://www.tmbroadcast.es/index.php/sdi-vs-ip-estado-actual/.

"Comparison between IP video protocols," [Online]. Available:
22] https://en.wikipedia.org/wiki/Network_Device_Interface#cite_note-smpte.org-8.

APB, "SDI vs IP: Current trends and future outcomes," [Online]. Available: https://apb-
23] news.com/sdi-vs-ip-current-trends-and-future-outcomes/.

24] MOG-Technologies, "CHIC," [Online]. Available: https://chic.mog-technologies.com/.

MOG-Technologies, "Activity C1. Framework de conceção, ingest e transporte de conteúdos de
25] elevada resolução em ambientes IP," 2018.

FFMPEG.org, "Documentação FFMPEG," [Online]. Available:
26] https://ffmpeg.org/documentation.html.

27] Riedel, "NMOS Explorer v1.3 Quick Guide," 2019.

"Session Description Protocol," [Online]. Available:
28] https://en.wikipedia.org/wiki/Session_Description_Protocol.

Interaction Design Foundation, [Online]. Available: https://public-media.interaction-
29] design.org/images/uploads/3c7d938af5ab0d5e6ac67be9536e3c47.jpeg.

30] "Github Node-RED," [Online]. Available: https://github.com/node-red/node-red.

31] "NMOS Technical Overview," [Online]. Available: http://amwa-tv.github.io/nmos/branches/master/NMOS_Technical_Overview.html.

32] "NMOS Discovery and Registration," [Online]. Available: https://amwa-tv.github.io/nmos-discovery-registration/tags/v1.2.1/docs/1.0._Overview.html.

33] Microsoft, "VPL Introduction," [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/microsoft-robotics/bb483088(v=msdn.10).