Orquestração multidomínio no *testbed* OpenRAN@Brasil

Daniel L. Feferman[1], Michael P. Hernandez[2], Wesley M. M. Santos[1], Michelle S. Chagas[1], Gustavo H. Araújo[2], Lucas B. Oliveira[2], Gustavo C. Lima[1]

[1]Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD) – Campinas, SP – Brasil

[2] Rede Nacional de Pesquisa (RNP) – Campinas, SP – Brasil

{dlazkani, wmartins, mchagas, gcorrea}@cpqd.com.br
{michael.hernandez, gustavo.araujo, lucas.oliveira}@rnp.br

Abstract. *This article describes the testbed that has been built in the OpenRAN@Brasil project targeting the challenge of a multidomain orchestration architecture. Several components of the testbed are described and the topology of the sites at CPqD and RNP. Furthermore, a comparison was presented between the various open source orchestration solutions available in the community, such as: Aether, Nephio and EMCO. Finally, the preliminary results of the multidomain orchestration proposal through a pre-testbed virtual environment are presented.*

Resumo. *O presente artigo descreve o testbed que tem sido construído dentro do projeto OpenRAN@Brasil visando os desafios de uma arquitetura com orquestração multidomínio. São descritos diversos componentes que irão compor o testbed e a topologia dos sites no CPqD e na RNP. Além disso, foi apresentada uma comparação entre as diversas soluções de orquestração de código aberto disponíveis na comunidade como Aether, Nephio e EMCO. Finalmente, são apresentados os resultados preliminares da proposta de orquestração multidomínio através de um ambiente virtual pré-testbed.*

1. Introdução

Na última década, as infraestruturas de rede se desenvolveram seguindo uma forte tendência em direção ao software em ambiente de nuvem, o que traz enormes benefícios, assim como diversos desafios. A softwarização facilita a programabilidade dos elementos de rede assim como a virtualização dos seus recursos, permitindo a alocação dinâmica e o particionamento da rede em fatias logicamente isoladas. Por sua vez, tais características impulsionam o desenvolvimento de componentes de software, principalmente controladores e orquestradores, que permitem gerenciar o ciclo de vida dessas fatias de rede, assim como das aplicações e serviços a elas associadas de forma totalmente programável [Mijumbi et al. 2016]. Essa orquestração quando realizada de forma completamente automatizada facilita enormemente a operação unificada da infraestrutura de rede, aumentando a flexibilidade, diminuindo a complexidade, reduzindo custos e evitando erros humanos [Checko et al. 2015]. Essa softwarização foi impulsionada pelo surgimento do paradigma SDN (Software-Defined Networking) [Kreutz et al. 2015].

Os orquestradores complementam os controladores SDN e contribuem na entrega de serviços através de múltiplos domínios de rede em uma infraestrutura *multi-vendor* ao abstrair a complexidade e capturar configurações da rede, dados operacionais e politicas em formato neutro. Dessa forma, os arquitetos de rede podem personalizar os modelos

de serviço integrados enquanto o orquestrador trata as diferenças na infraestrutura subjacente. Como resultado, as operadoras de rede não precisam aprender CLIs complexas de vários fornecedores e os serviços de rede podem ser implantados de forma ágil. Os orquestradores evitam as dependências tecnológicas e de fornecedores para a proteção do investimento. Eles permitem a fácil integração e automação com portais de autoatendimento dos clientes, aplicações Web de terceiros e OSS/BSS por meio de APIs REST e não possuem parâmetros *hard-coded* para o lançamento de novos tipos de dispositivos e serviços. Além disso, reduzem OpEx e CapEx e melhoram drasticamente a estrutura de custos por meio da independência parcial ou total do fornecedor e da automação de processos com chaves de redes multi-vendor por meio de APIs para unificar o gerenciamento de rede tradicional, a programação de rede aberta no estilo SDN e a orquestração do ciclo de vida do serviço [Mustafiz et al. 2016]. Portanto, o uso de orquestradores reduz a complexidade geral da rede, eliminando silos operacionais e unificando todas as funções de gerenciamento em redes fim-a-fim, de vários fornecedores e de várias camadas em uma plataforma única.

O projeto OpenRAN@Brasil[1] desde 2021 vem sendo desenvolvido em uma parceria entre o CPQD[2] e a RNP (Rede Nacional de Ensino e Pesquisa)[3] com o apoio de diversas universidades como: UFPA (Universidade Federal do Pará) e UNICAMP (Universidade de Campinas). Além disso, dentre suas metas, uma delas é a construção de um *testbed* usando infraestrutura de nuvem, em sua maioria em *Kubernetes* orquestrada através de uma solução multidomínio. Dessa forma, o presente artigo detalha as diversas soluções utilizadas no *testbed*[4] no sentido de orquestração em ambiente multidomínio.

Este artigo está organizado da seguinte forma: A Seção 2 apresenta uma descrição dos equipamentos e os domínios tecnológicos que será formado o *testbed*. Na Seção 3, é apresentada uma tabela comparativa do estado da arte relativo aos softwares de orquestração e a arquitetura proposta para o testbed. Na Seção 4, são apresentados os resultados da implantação virtual e dos testes já realizados. Por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. *Testbed* do projeto OpenRAN@Brasil

O *testbed* do projeto OpenRAN@Brasil tem sua topologia apresentada na Figura 1. A lista de equipamentos que irão compor o *testbed* é apresentada na Figura 2, onde constam diversos detalhes, dentre eles: tipos, *labels*, funções dos equipamentos, fabricantes, modelos e quantidades em cada *site*. Ambos os *sites* possuem um conjunto de 7 servidores categorizados em uma arquitetura espelhada. Dessa forma, a topologia divide o ambiente em dois clusters Kubernetes, um de desenvolvimento, usado para evolução da arquitetura e outro de produção, com garantia de maior estabilidade frente ao anterior.

[1]https://www.rnp.br/projetos/openranbrasil

[2]https://www.cpqd.com.br/

[3]https://www.rnp.br/

[4]Atualmente o testbed está sendo realizado em um cenário menor em ambiente controlado pela RNP. Entretanto, ainda em 2023 é esperado o ambiente definitivo concluído, tendo como resultado diversos servidores virtualizando máquinas e uma arquitetura kubernetes considerando os sites do CPqD (Campinas) e RNP (Rio de Janeiro)

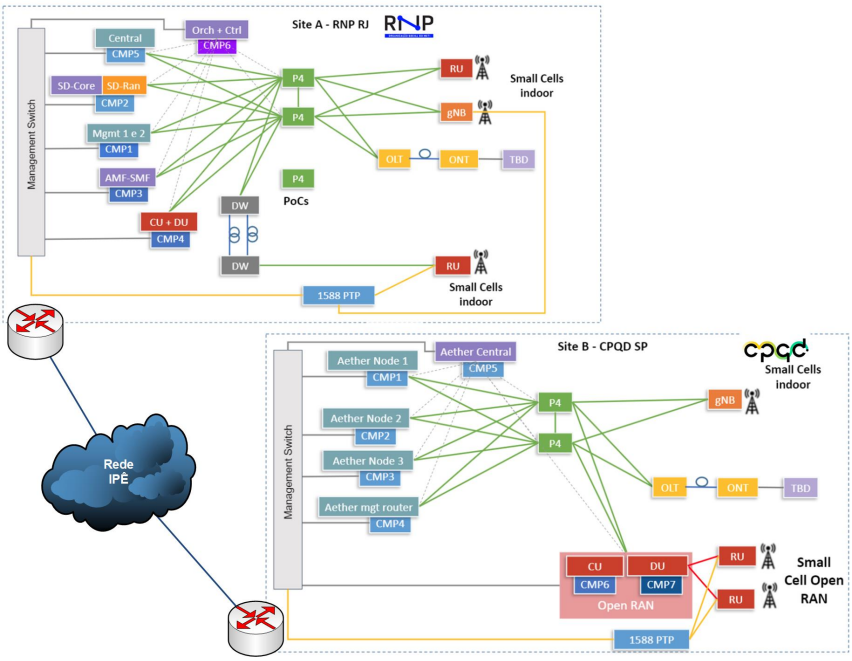


Figura 1. Topologia do *testbed* do projeto OpenRAN@Brasil que será disponibilizado para a comunidade no terceiro trimestre de 2023.

Equipamentos					Nós por Destino	
Tipo	Label no diagrama	Funções	Fabricante	Modelo	RNP	CPqD
Compute Tipo 01	CMP1	Bastion, Aether Central Node, SmartEdge RNP	Supermicro	SYS-220U-MTNR	3	2
Compute Tipo 02	CMP2	CU+DU (RNP), DU (CPqD)	Supermicro	SYS-220U-MTNR	2	2
Compute Tipo 03	CMP3	Aether Compute Server	Supermicro	SYS-220U-MTNR	1	1
Compute Tipo 04	CMP4	CU (CPqD)	Supermicro	SYS-110P-FRN2T	0	1
Compute Tipo 05	CMP5	Orquestração RNP	Supermicro	SYS-220U-MTNR	1	1
Switches P4	P4	UPF	Edge-Core	DCS801	2	2
RU	RU	Antena OpenRAN	Foxxcon	Indoor RRU RPQN-78	3	3
Sw Gerencia	Management Switch	Switch Gerencia	UFiSpace	S9600-72XC	1	1
OLT Combo	OLT	OLT GPON e XGS-PON	Radisys	RLT-1600X	0	1
OLT XGS-PON	OLT	OLT XGS-PON	Edge-Core	ASXvOLT16	1	0
ONT GPON	ONT	ONT GPON 4x1Gb, 2xPOT e WiFi	Radisys	PM-4264	0	4
ONT XGS-PON	ONT	ONT XGSPON 1x10Gb, 4x1Gb, 2xPOT e WiFi	CIG	XG99-XS	4	4
GrandMaster	1588 PTP	Sincronismo de Relógio	Fibrolan	FalcomRX	1	1
Cassini	DW	DWDM Programável	Edge-Core	AS7716-24SC	2	0

Figura 2. Lista de equipamentos do *testbed*.

3. Arquitetura de orquestração

Em nossa pesquisa pela solução de orquestração para nosso testbed, encontramos alguns projetos. Entretanto, verificamos que nenhum deles atende sozinho o quesito de orquestração multidomínio. Sendo assim, a Tabela 1 compara os diferentes projetos disponíveis na comunidade com suas características.

Considerando a Tabela 1, observa-se que o ONAP embora seja um projeto de grande relevância na comunidade, possui requisitos mínimos robustos, como: 224 GB RAM, 112 vCPUs e 160GB de armazenamento, tornando-o uma solução onerosa ao projeto. Adicionalmente, o ONAP tem sofrido para receber novas funcionalidades e apoio da indústria, além de ser o único, das ferramentas encontradas, a utilizar TOSCA como linguagem para modelagem de dados. Portanto, nesse momento, a ferramenta não é considerada no testbed.

A especificação ETSI MANO assim como a sua implementação de referência [OSM 2023], foram desenvolvidas no contexto de máquinas virtuais (VM). Com o Ku-

Tabela 1. Comparação das diferentes soluções de orquestração.

Projetos	Organização	Casos de uso	Requerimentos	Maturidade	Complexidade	Modelamento de dados
[OSM 2023]	ETSI	Service chaining 5G Slicing MEC	Alto	Alto	Médio	YANG
[ONAP 2023]	Linux Foundation	5G vCPE VoLTE CCVPN	Alto	Médio	Alto	TOSCA
[Nephio 2023]	Linux Foundation	5G CNF MEC	Médio	Baixo	Médio	YANG
[Aether 2023]	ONF	MEC 5G	Baixo	Médio	Baixo	YANG
[EMCO 2023]	Linux Foundation	5G MEC CNF	Médio	Alto	Médio	YANG

bernetes se tornando a plataforma dominante para orquestração de contêineres, uma especificação de orquestração diretamente em termos nativos do Kubernetes forneceria simplificação e permitiria aproveitar os recursos e extensões do Kubernetes para atender aos requisitos MANO. Na ausência de tais especificações mapeadas diretamente, várias maneiras de interpretar os requisitos MANO nos contextos nativos da nuvem ou em containers são possíveis. Nesse sentido, vários projetos abordam os desafios de definir um sistema de orquestração em contextos nativos de nuvem e/ou containers. Diversos projetos de código aberto com Linux como o Nephio e EMCO apontam para a necessidade de tal trabalho. Embora o OSM também tenha dado passos na integração com o Kubernetes, ainda não há maturidade suficiente para ser considerado como uma peça na orquestração multidomínio dentro do OpenRAN@Brasil.

O [Aether 2023] é um projeto desenvolvido pela ONF que vem sendo desenvolvido há alguns anos através do projeto OMEC e tem baixos requisitos. Dessa forma, é possível fazer a instalação da solução *Aether-in-a-box* em um notebook com uma baixa complexidade para execução. O Aether tem como solução para orquestração o nível de *core*. Adicionalmente, o Aether utiliza a linguagem de modelamento de dados YANG, bastante difundida no mercado.

O [EMCO 2023] foi iniciado pela Intel como parte do projeto *Smart Edge* e posteriormente aberto à comunidade, tornando-se parte da Linux Foundation. Portanto, o mesmo apresenta certo grau de maturidade. Para utilizar o EMCO é necessário um cluster Kubernetes e também Helm *Charts* para realizar a sua instalação. Em um cenário pré-testbed, uma arquitetura com cluster Central e de borda foi contruída. Nesse cenário, o EMCO se mostrou uma solução pertinente para a orquestração multidomínio. Sendo assim, a solução deverá ser incorporada ao ambiente do testbed.

O [Nephio 2023] é um projeto novo que assim como o EMCO visa realizar a orquestração a nível de containers através do Kubernetes. Idealizado ainda em 2022, sua primeira *release* (R1) ainda está em desenvolvimento. Porém, algumas provas de conceito do Nephio já foram implementadas no cenário *pré-testbed*, demonstrando-se uma ferramenta promissora, visto que grande parte de sua arquitetura já está desenvolvida. Adicionalmente, o *core* 5G usado de referência para a prova de conceito do Nephio atualmente é o [Free5GC 2023] o qual é a base do módulo 5G no Aether facilitando a integração entre

ambos projetos.

Finalmente, no estado da arte analisado no contexto de orquestração multi-domínio, não foi identificada uma única solução de código aberto capaz de atuar em todos os domínios do *testbed*. Sendo assim, o uso de uma solução considerando uma API Gateway é necessária visando proporcionar uma interface como ponto de acesso único aos serviços oferecidos pelo *testbed*. Dessa forma, ao usar o *API gateway*, uma camada entre os clientes (usuários ou aplicações) e a coleção de APIs dos serviços *back-end* disponíveis é adicionada a arquitetura. Tal camada realiza as requisições internamente em nome do cliente, assim abstraindo a complexidade dos serviços e ocultando a sua origem para quem está fora do ambiente interno. Portanto, é considerado o uso da ferramenta Kong API Gateway como solução para a integração das múltiplas APIs do ambiente de testbed. Sendo assim, a Figura 3 ilustra a arquitetura final, onde os elementos em lilás representam os orquestradores para o testbed e sua camada de atuação.

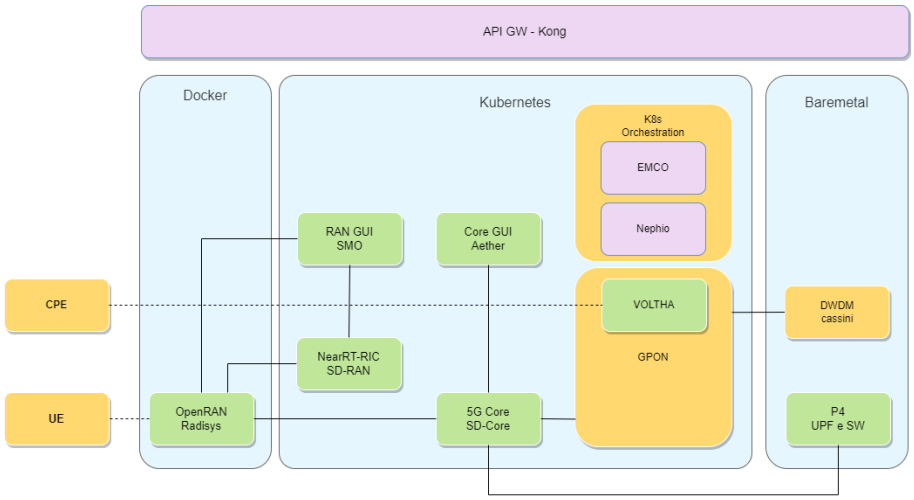


Figura 3. Arquitetura do do testbed considerando os múltiplos domínios e as soluções de orquestração.

O *API gateway* é responsável por encaminhar as requisições e agregar funcionalidades a infraestrutura, como controle de acesso, monitoramento, limitação de uso, etc. A segurança proporcionada pelo *API gateway* o coloca na fase de controle, quando se observa o ciclo de vida de uma API (criação, controle e consumo), pois não expõe o ambiente interno e seus dados ao ambiente externo. Em consequência, possibilita os clientes realizarem requisições mais simples, passando a responsabilidade de encaminhar e aplicar políticas de acesso, interação e autenticação ao *API gateway*. Dessa forma, o *API gateway* pode ser adicionado à frente dos serviços para interceptar as requisições, seguindo o esquema da Figura 4.

Por atuar como a única porta de entrada para todos os serviços, as requisições dos clientes e suas respostas são concentradas nessa camada. Logo, ao receber uma requisição, faz-se necessário que o *API gateway* saiba a qual serviço deve encaminhar. Dessa forma, alguns conceitos que auxiliam no entendimento da ferramenta são utilizados e listados a seguir:

- **Clientes:** usuários ou aplicações que consomem recursos, sejam eles dados ou funcionalidades dos serviços disponíveis no *back-end*.

- **Serviços:** coleção de APIs ou microsserviços abstraídas pelo *API gateway*, ou seja, é o destino final das requisições realizadas pelos clientes.
- **Rotas:** caminho responsável por alcançar o serviço desejado. A sua implementação ocorre na proporção de 1:n com os serviços, ou seja, para cada serviço podem existir diversas rotas.

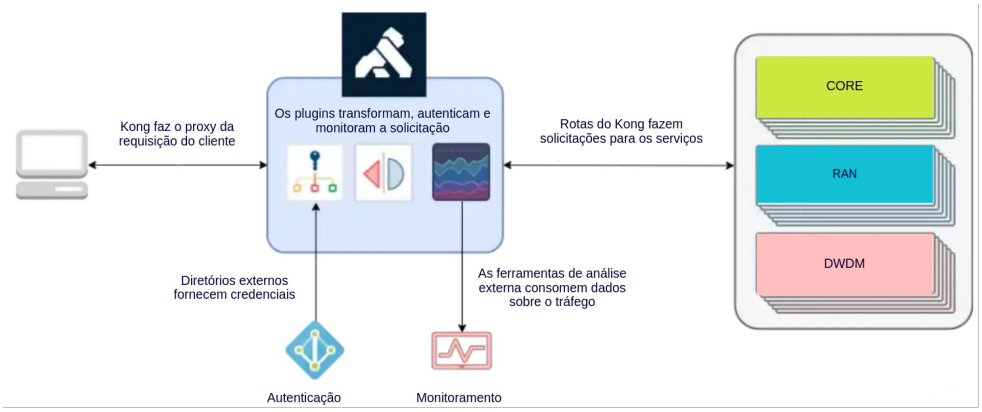


Figura 4. Padrão de funcionamento de um *API gateway*.

As requisições são feitas ao *gateway* de forma transparente para o cliente. Dessa forma, o *API gateway* pode manusear os dados através de duas abordagens: fazendo um *proxy*/encaminhando para o serviço correto ou então, manipulando a requisição original, podendo invocar múltiplos serviços, alterar protocolos, definir fluxos de trabalho, etc. Dessa forma, as requisições passam a exigir alguns padrões que podem ser utilizados para determinar como serão tratadas e respondidas de maneira apropriada.

4. Resultados

O projeto de testbed OpenRAN@Brasil propõe a integração dos domínios sem fio, óptico e de pacotes por meio de controladores SDN especializados em cada um desses domínios tecnológicos e assim, controlar de forma integrada os serviços de rede através da orquestração. Sendo assim, o orquestrador é responsável por realizar o gerenciamento integrado e inteligente dos recursos e serviços envolvidos em diferentes cenários de rede. Abaixo são abordadas as soluções estudadas nessa seção sob essa ótica.

4.1. Solução de core - Aether

A solução Aether-in-a-box (AiaB) provê uma instalação simplificada do Aether com SD-Core permitindo efetuar a orquestração na camada do core 5G. Os requisitos mínimos do AiaB são: Ubuntu 18.04 com instalação limpa, Kernel 4.15, CPU Haswell ou mais recente, ao menos 4 CPUs e 12 GB de memória RAM. A Figura 5 demonstra os componentes internos da plataforma *Aether Management Platform* (AMP).

4.1.1. SD-Core

O SD-Core é o elemento a ser orquestrado pelo Aether. Ele é um core de redes móveis 5G otimizado para implantação em nuvem pública, ideal para redes privadas e seguindo a padronização 3GPP. A Figura 6 ilustra a arquitetura do SD-Core.

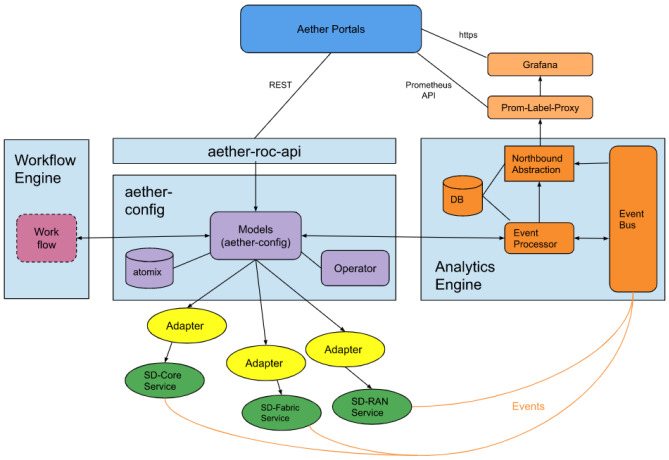


Figura 5. Arquitetura do Aether Management Platform.

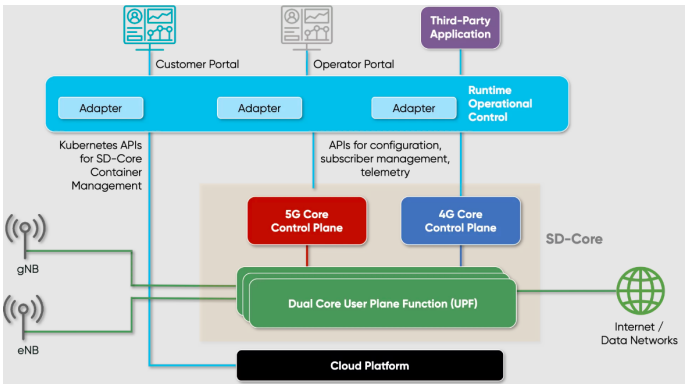


Figura 6. Arquitetura do SD-Core.

Com o intuito de aprofundar nosso conhecimentos e validar as funcionalidades descritas na documentação do Aether como orquestrador do SD-Core, foi criado um ambiente mínimo para viabilizar os testes, utilizando uma máquina virtual presente em um servidor localizado no data-center da RNP. Dessa forma, utilizamos um servidor que possui como características: Intel(R) Xeon(R) CPU E5-2630 v3 2.40GHz, 8 vCPUs, 16GB de memória Ram e 55GB de HD. Nesse único servidor foram instaladas as soluções *Aether-in-a-box* versão 2.0, contendo o orquestrador Aether e SD-Core na versão 5G, além do UERANSIM, um emulador de gNB e UE. Dessa forma, todas as ferramentas foram usadas em um ambiente conteinerizado utilizando kubernetes versão v1.23.4+rke2r1.

Para realizar os testes foi necessário instalar o *aether-in-a-box* em containers com um cluster Kubernetes e realizar sua integração ao emulador UERANSIM. A Figura 7 demonstra a topologia final do ambiente criado para os testes, utilizando máquina virtual instalada dentro do data-center da RNP.

Os testes de funcionalidades suportados pela solução foram divididos em tópicos para a melhor organização e entendimento. Sendo eles os seguintes: gestão de assinantes e dispositivos, configuração por API, monitoramento utilizando interface gráfica do Aether, gerenciamento de *slices*, QoS e gerenciamento de aplicações. Os testes de gestão de assinantes e dispositivos estão relacionados a funcionalidades de provisionamento de usuários, administração dos mesmos e seus perfis.

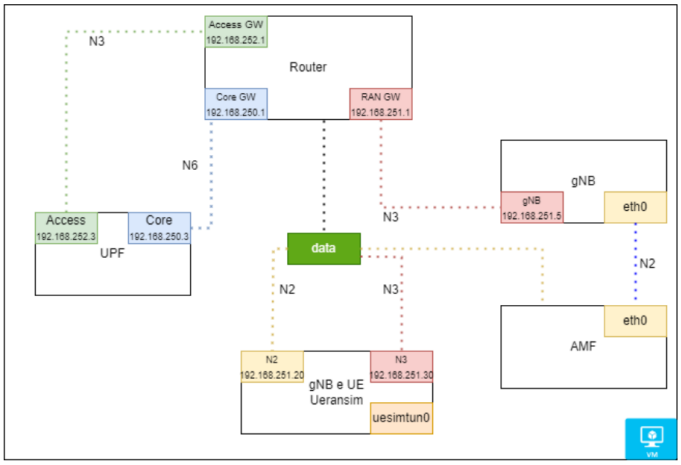


Figura 7. Topologia do ambiente de testes instalado no servidor RNP.

Outra maneira de criar novos assinantes atrelados a chaves autenticáveis ou deletá-los é via API REST, utilizando uma ferramenta externa para envio da solicitação. Nesse caso, foi utilizado o Postman para realizar os testes a partir de um servidor externo, apontando para IP do servidor do Aether, na porta do serviço "webui" (5000) utilizando protocolo TCP. A Figura 8 demonstra um exemplo de solicitação API do teste descrito. Além disso, foram testadas a utilização das APIs fornecidas através do endereço *http://localhost:8181/aether-2.1.0-openapi3.yaml*, que tem por objetivo configurar o portal do Aether ROC diretamente via API, populando as informações sem precisar usar a interface gráfica do portal. Adicionalmente, também foi possível realizar pedidos de "GET" no Aether para coletar informações do ambiente em tempo real. Por último, as solicitações via API REST são enviadas para o serviço "aether-roc-api" que atua na porta TCP 8181, conforme exemplificado na Figura 9.

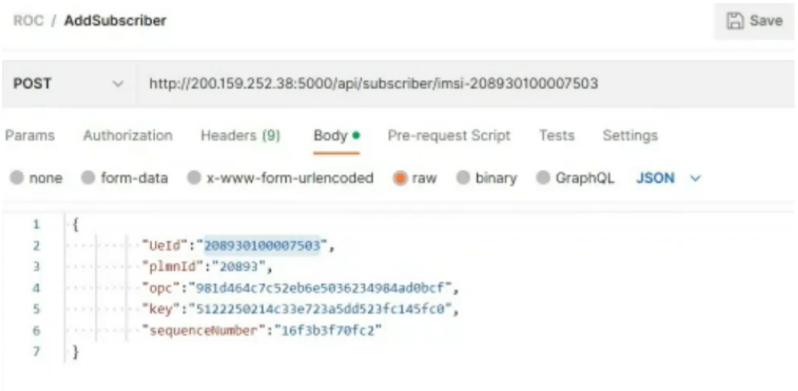


Figura 8. Solicitação de RestAPI enviada via Postman para o serviço webui do servidor do Aether.

Através do portal do Aether ROC (*Runtime Operation Control*) é possível criar as configurações relacionadas ao perfil do usuário usando os campos: "device group", "simcard", "device" e "slice". Porém, é importante ressaltar que o passo de atrelar um assinante as suas chaves autenticáveis somente pode ser feito pelos métodos descritos anteriormente, ou seja, via "simapp" ou solicitação API REST.

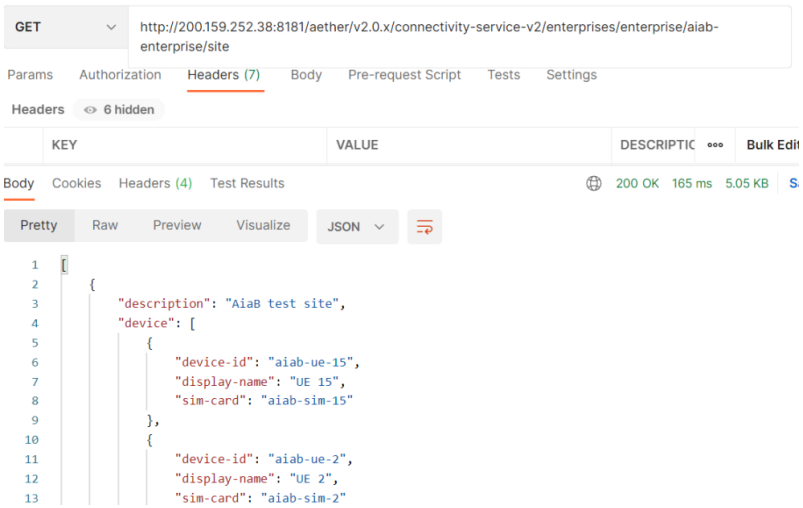


Figura 9. Exemplo de solicitação API enviada para o servidor utilizando a porta 8181 do serviço aether-roc-api.

4.2. API Gateway

Cada domínio tecnológico possui um padrão de configuração, comunicação e tratamento de dados. Visando diminuir a complexidade, o *API gateway* pode ser adicionado entre o cliente e os controladores SDN dos diferentes domínios. Dessa maneira, as requisições são direcionadas ao *gateway* que se torna responsável por tratar e redirecionar aos domínios corretos.

A abordagem utilizando o *API gateway* pode ocorrer adicionando o *Kong Gateway* entre o cliente e os controladores de cada domínio para abstrair a complexidade do ambiente em diversas esferas, a Figura 10 ilustra a perspectiva no contexto do projeto. Cada controlador disponibiliza suas APIs RESTFul que entregam uma interface administrativa que possibilita a gerência do domínio controlado. O *Kong Gateway* entende essas APIs como locais para o qual serão redirecionadas as requisições.

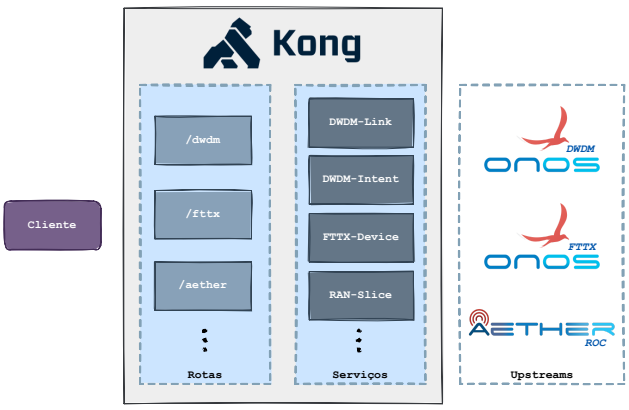
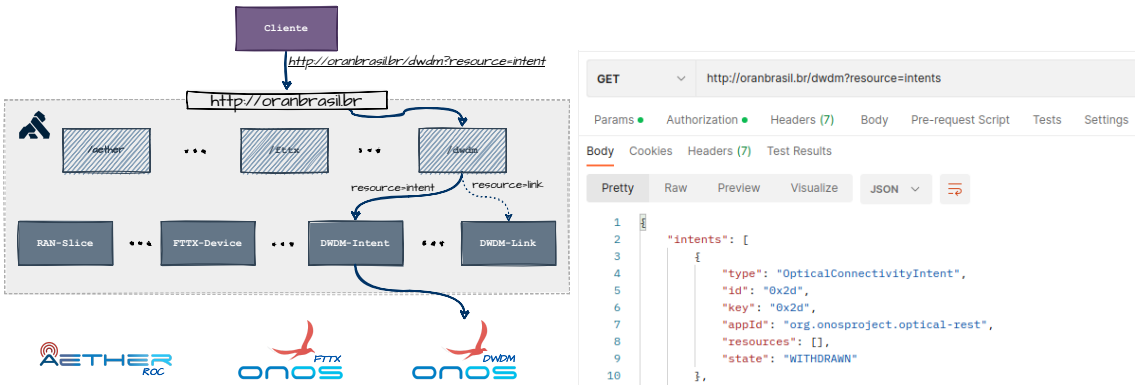


Figura 10. Arquitetura do *Kong Gateway* aplicada aos elementos do projeto.

Os serviços e as rotas são configurados de maneira coordenada para definir o encaminhamento que as solicitações e respostas terão pelo sistema. Em tal caso, as rotas são adicionadas aos serviços para permitir o acesso às APIs subjacentes. Basicamente,

no *Kong Gateway*, as rotas possuem nome, caminho ou caminhos e são mapeadas a um serviço existente, e consequentemente às APIs expostas.

O cliente que desejar consumir as APIs, seja para provisionar recursos, seja para recuperar informações dos domínios, passam a fazer requisições ao *Kong Gateway* e este é responsável por encaminhar ao domínio correto. A requisição do cliente é formada por alguns atributos como: domínio e os recursos desejados. Então, ao receber a requisição, o Kong a desmembra e verifica o domínio baseado na rota cadastrada e o serviço requerido baseado nos parâmetros passados. Dessa forma, esse conjunto de informações levam até à API correta do controlador do domínio desejado. A Figura 11(a) reforça o percurso interno e as verificações realizadas pela ferramenta para responder a requisição do cliente, enquanto que a Figura 11(b) exibe a resposta obtida.



(a) Processamento da requisição no *Kong Gateway*. (b) Resposta referente a requisição do cliente.

Figura 11. Método de utilização da *API Gateway* no contexto do projeto.

4.3. Integração Multidomínio

Para realizar a integração entre múltiplos domínios tecnológicos foi proposto um experimento de transferência de dados entre um UE (*User Equipament*) e um servidor de conteúdo. Para realizar a transferência, os dados devem trafegar entre no domínio *Mobile* (Core/RAN) e o domínio DWDM. A Figura 13 ilustra as diferentes camadas presentes neste experimento. A camada de "Domínio Tecnológico" exemplifica o cenário propostos e os domínios envolvidos no experimento. A camada "Plataforma" ilustra as plataformas utilizadas para representar cada domínio tecnológico: (*i*) UERANSIM: utilizado para simular aspectos da RAN e UEs; (*ii*) Aether-in-a-box: para representar o core da rede mobile; e (*ii*) CNetLab: para representação do domínio DWDM. Por fim, a camada "Elementos" representa os detalhes técnicos utilizados para a construção deste cenário (eg., *switches*, controladores, elementos de software).

No domínio RAN foi utilizado o UERANSIM para simular os UEs e gNodeBs. Dessa forma, um UE neste domínio tecnológico, deve se comunicar com um servidor de conteúdo presente no Domínio DWDM. Para representar o domínio CORE foi utilizado o *Aether-in-a-box* (AiaB). Como descrito na seção 4.1.1, AiaB fornece uma maneira fácil de realizar a implantação de seus componentes, permitindo a execução de testes básicos. Além disso, para a representação do Domínio DWDM foi utilizado o laboratório CNetLab. Esse laboratório permite uma maneira fácil instanciar *switches Stratum* e *cassini*,

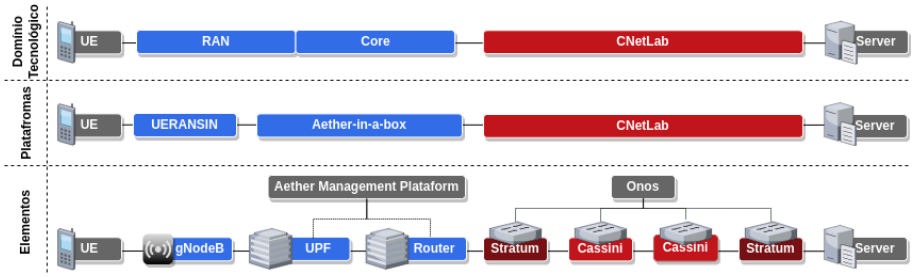


Figura 12. Arquitetura da integração Multidomínio.

terminal de usuário, controlador ONOS e enlaces virtuais entre os *switches*. Além disso, através das APIs de controle, que são expostas pelo controlador ONOS é possível configurar circuitos e verificar estatísticas de cada um dos *switches*.

No domínio Core é necessária uma ligação entre o componente Router e um switch stratum na topologia DWDM. No domínio DWDM a topologia é composta por dois switches Stratum, dois switches cassini, um controlador ONOS e um *host* que funciona como servidor de conteúdo. Os switches estão conectados em linha de acordo com o que é mostrado na Figura 13 na camada "Elementos". Dessa forma, para controle dos switches foi utilizado o controlador ONOS com as seguintes aplicações habilitadas: *optical rest*, *netconf subsystem*, *host provider*, *network link provider*, *LLDP provider*, *proxy arp* e *forward*. Além disso, para configurar os switches foi utilizada a aplicação de *Intents* no ONOS. A aplicação cria regras nas tabelas dos switches permitindo que um circuito seja criado em todos os elementos da topologia. Na Figura 13, pode-se observar os parâmetros necessários para criar um circuito no domínio DWDM. Adicionalmente, é necessário informar o ponto de início e fim do circuito, as portas necessárias e se será unidirecional ou bidirecional. Por último, o controlador ONOS configurará as regras nas tabelas de fluxos dos switches stratum e cassini e será possível uma comunicação entre o UE e o servidor de conteúdo.

```
{
    "appId": "org.onosproject.optical-rest",
    "ingressPoint": {
      "device": "netconf:172.17.0.6:830",
      "port": "201"
    },
    "egressPoint": {
      "device": "netconf:172.17.0.5:830",
      "port": "201"
    },
    "bidirectional": "true",
    "suggestedPath": {
      "links": [
        {
          "src": "netconf:172.17.0.6:830/201",
          "dst": "netconf:172.17.0.5:830/201"
        }
      ]
    }
}
```

Figura 13. Parâmetros para criação de circuitos através da aplicação de Intents do ONOS.

5. Conclusões e trabalhos futuros

Neste trabalho foi apresentada a visão geral do *testbed* OpenRAN@Brasil. Nele foi realizada uma comparação entre diversas soluções para orquestração e foi proposta uma arquitetura de orquestração multidomínio.

Pode-se concluir que a linha de pesquisa relacionada a orquestração multidomínio é desafiadora e ainda está sendo amplamente estudada pela comunidade científica e o mercado dada a sua complexidade. Sendo assim, não foi identificada uma solução única de código aberto capaz de orquetrar todos os elementos de nosso *testbed*, mas sim um conjunto de soluções que atuam em camadas independentes porém integradas através de uma API centralizada. Foi selecionado o Aether como solução de orquestração do core 5G, o Kong como centralizador das APIs dos diversos domínios do *testbed* e consideramos o uso de um orquestrador Kubernetes (i.e., EMCO, Nephio).

Como trabalhos futuros pretende-se contribuir na implantação do *testbed* físico assim como na integração com a solução de orquestração multidomínio proposta. Além disso, serão relizados estudos sobre a viabilidade de estender a solução Nephio visando orquestrar múltiplos domínios tecnológicos.

Referências

Aether (2023). Open Networking Foundation (ONF). Disponível em: `https://opennetworking.org/aether/`. Acesso em junho de 2023.

Checko, A., Christiansen, H. L., Yan, Y., Scolari, L., Kardaras, G., Berger, M. S., and Dittmann, L. (2015). Cloud ran for mobile networks—a technology overview. *IEEE Communications Surveys Tutorials*, 17(1):405–426.

EMCO (2023). Edge Multi-Cluster Orchestrator. Linux Foundation. Disponível em: `https://project-emco.io/`. Acesso em junho de 2023.

Free5GC (2023). Free 5G Core Network. Disponível em: `https://www.free5gc.org/`. Acesso em junho de 2023.

Kreutz, D., Ramos, F., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Mijumbi, R., Serrat, J., Gorricho, J.-l., Latre, S., Charalambides, M., and Lopez, D. (2016). Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105.

Mustafiz, S., Palma, F., Toeroe, M., and Khendek, F. (2016). A network service design and deployment process for nfv systems. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 131–139, Los Alamitos, CA, USA. IEEE Computer Society.

Nephio (2023). Cloud Native Network Automation. Linux Foundation. Disponível em: `https://nephio.org/`. Acesso em junho de 2023.

ONAP (2023). Open Network Automation Plataform. Linux Foundation. Disponível em: `https://www.onap.org/`. Acesso em junho de 2023.

OSM (2023). Open Source Mano. ETSI. Disponível em: `https://osm.etsi.org/`. Acesso em junho de 2023.