OCClient: Ferramenta para Gerenciamento integrado de Dispositivos em Redes SDN

José Ferreira[2], Murilo Silva[1], Matheus Gomes[1], Victor Leite[1], Dionísio Fares[2], Antônio Abelém[12]

[1] Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal do Pará (UFPA) – Belém – PA – Brasil

[2]Instituto de Ciências Exatas e Naturais (ICEN)
Universidade Federal do Pará (UFPA) – Belém – PA – Brasil

{murilosilva,matheus.cordovil}@itec.ufpa.br

victor.leite@ig.ufpa.br, dionisiofaressilva@gmail.com

{joseferreira,abelem}@ufpa.br

Abstract. *While network administrators may typically handle integration with APIs, doing so, especially with ONOS, SDN controller, which lacks an SDK client, can be time and resource-intensive. To address this, OCClient was developed as a support tool, offering a more streamlined approach to interacting with the controller. OCClient's binary and compatibility across most environments simplify adoption, especially within automated management flows like GitOps. For the testing scenario, a laboratory was provisioned with devices emulated by CNETLAB and a controller, both without relationships. Finally, a JSON file containing device declarations and their respective links was used to apply device registration settings to the controller. Subsequently, the tool was also used to remove links and register devices from the controller. Lastly, a comparison was made between applying the settings through the tool or through direct implementation using a programming language, considering the requirements for the development of the tool.*

Resumo. *A tarefa de integração com APIs pode ser onerosa, tanto em tempo como por conta de custo com pessoal. No caso de Redes Definidas por Software (SDN), que utilizam o controlador ONOS, essa situação é especialmente problemática já que ele não possui um SDK. Este trabalho apresenta uma solução para gerenciamento de dispositivos, denominada OCClient. Inicialmente a solução foi desenvolvida para ser uma ferramenta de apoio ao grupo de trabalho da UFPA, durante a fase 1, do programa OpenRAN@Brasil. Posteriormente, notou-se um interesse da comunidade por uma ferramenta que pudesse auxiliar na interação com o controlador. Por tratar-se de um binário, pode ser executado na maioria dos ambientes, sem a necessidade de instalação, facilitando a adoção em fluxos automatizados de gerenciamento de configuração de redes, tal como acontece na abordagem GitOps. Para os testes foi provisionado um laboratório com dispositivos emulados pelo CNETLAB e um controlador, um arquivo JSON, contendo as declarações dos dispositivos e respectivos links foi utilizado para aplicar as configurações de registro dos dispositivos no controlador. Posteriormente, a ferramenta também foi utilizada para realizar a remoção dos dispositivos do controlador. E por fim, foi realizada uma comparação entre*

aplicar as configurações através da ferramenta ou por meio de uma implementação direta utilizando utilizando uma linguagem de programação.

1. Introdução

As possibilidades apresentadas pelas redes SDN como, análise de estado da rede, obtenção de informações sobre sua topologia, descoberta de dispositivos conectados, definição das configurações de rede e políticas de roteamento, corroboram para a configuração de uma nova perspectiva sobre o gerenciamento de dispositivos de rede, dada a simplificação proporcionada pela sua adoção [Liatifis et al. 2023]. Com o desacoplamento do plano de dados do plano de controle o gerenciamento do dispositivos, assim como suas funções de redes, podem ser abordadas como software. Usualmente, essa possibilidade é apresentada ao usuário na forma de APIs, cuja arquitetura depende principalmente da latência tolerada em seu emprego. Aplicações com requisitos de maior desempenho, são usualmente associadas ao uso de *gRPC/http2*, enquanto no restante das aplicações existe predominância no uso de *REST/http*, utilizada pelo *ONOS* [ONF 2023].

O objetivo do projeto é desenvolver um método de interação com o controlador, cuja carga de conhecimento necessária seja menor, em comparação com o método padrão via API REST/http, abrangendo inicialmente as principais funcionalidades (inserção, listagem e remoção de dispositivos). A flexibilidade apresentada por uma abordagem programática é extremamente útil para casos específicos, mas é uma barreira extra para os casos comuns de uso. O esforço de desenvolvimento dessas operações comuns é repetitivo, tendo em vista que todos os que escolherem utilizar o *ONOS* como controlador deverão passar pelos mesmos desafios. Diante desse cenário, a solução foi desenvolvida como um projeto de apoio ao grupo de trabalho da UFPA no programa OpenRAN@Brasil [RNP et al. 2021] e posteriormente apresentada a outros grupos do programa, que demonstraram bastante interesse em uma ferramenta do tipo.

O documento está organizado da seguinte forma: a Seção 2 apresenta a metodologia utilizada no desenvolvimento do trabalho. A Seção 3 apresenta os detalhes que permearam a concepção do projeto, filosofia de desenvolvimento adotada, inspirações e declarativa das ferramentas de apoio. A Seção 4 apresenta uma demonstração de uso da solução e aborda outras aplicações para a ferramenta. A Seção 5 apresenta os resultados obtidos e a análise destes. Por fim, Seção 6 é destinada às conclusões e sugere alguns trabalhos futuros.

2. Metodologia

Para o desenvolvimento foram levantados os seguintes pontos: as operações básicas, que todos os usuários do *ONOS* precisariam realizar; conhecimentos necessários para realizar a integração com a API do controlador, essencialmente o conhecimento em uma linguagem de programação, JSON e interação com APIs REST/http; a forma como a ferramenta seria disponibilizada, que deveria ser o mais simples possível, de preferência um binário; e qual seria o a filosofia de desenvolvimento mais adequada para proporcionar a familiaridade desejada para o utilizador, haja vista que seria uma ferramenta de linha de comando. Além disso, para a realização dos testes durante o desenvolvimento seria necessário uma pilha de ferramentas para emular os dispositivos e o controlador.

O plano de controle foi implementado com base na distribuição *2.5.9.ORANBR* do ONOS [Silva et al. 2023a], desenvolvida pelo grupo de trabalho da UFPA durante a fase 1 do programa OpenRAN@Brasil, e os dispositivos do plano de dados provisionados com auxilio do emulador *CNETLAB* [RNP 2023], ferramenta desenvolvida durante o projeto SDN Multicamadas da Rede Nacional de Ensino e Pesquisa (RNP), que emula dispositivos através de *containers*.

A linguagem *Go* foi escolhida, com base nos critérios de disponibilização em binário, forte adoção na comunidade, e utilizamos a biblioteca *Cobra* para auxiliar no desenvolvimento de uma interface amigável que pudesse ter uma baixa curva de aprendizado para, principalmente mas não exclusivamente, usuários de sistemas *UNIX-like*. Essas ferramentas de desenvolvimento foram selecionadas focando tanto no usuário final, quanto nos colaboradores do projeto, oferecendo um ambiente com ferramentas de grande adoção, propicio para colaborações futuras.

O ambiente de testes consiste em um controlador provisionado fora do *CNETLAB*, e dispositivos dos diferentes domínios tecnológicos, *Stratum* e *Cassini* (domínio óptico) [Silva et al. 2023b], provisionados pelo *CNETLAB*. O objetivo é realizar o ingresso desses dispositivos na topologia do ONOS através do *OCClient*, ferramenta apresentada por esse trabalho, analogamente como acontece com os recursos do *Kubernetes* declarados via *yamls* e a ferramenta `kubectl`.

Os testes foram realizados em cinco etapas, o provisionamento dos dispositivos; ativação das aplicações necessárias para cada domínio tecnológico; ingresso dos dispositivos na topologia do controlador; listagem dos dispositivos ativos; e remoção de dispositivos.

Para realizar o registro dos dispositivos é necessário um *payload* específico, assim como acontece com os recursos do *Kubernetes* (*Deployments*, *Ingresses* e *Services*). Além dos dispositivos, ainda é necessário registrar os *links* entre os dispositivos, e tal qual acontece com o registro de dispositivos, cada tipo de *link* tem um *payload* específico.

Com relação a classificação desta pesquisa, o principal método científico utilizado é o dedutivo, no qual busca-se um gerenciamento inteligente da tecnologia de redes definidas por software. Quanto a sua abordagem, esta pesquisa apresenta uma metodologia qualitativa, definidas a partir da carga de conhecimentos necessários para realizar as operações com a ferramenta e o método padrão, via API *REST/http*. Com relação à natureza, este estudo enquadra-se como uma pesquisa aplicada, tendo em vista que é feita a implantação dos dispositivos do plano de dados e do plano de controle, bem como, realizado o experimento de conexão envolvendo estes dispositivos.

3. Funcionamento Lógico da Solução

A ferramenta segue a filosofia *kiss (Keep is simple, stupid!)* e *UNIX* de boas práticas para o desenvolvimento de *CLIs* (*command line interfaces*), demonstrando seu compromisso com a usabilidade, entregando ao usuário um ambiente familiar desde o primeiro contato.

Para tornar isso possível, a abordagem escolhida foi disponibilizar um binário que pudesse ser executado nos diversos sistemas operacionais, sem que houvesse a necessidade de retrabalho extensivo para o desenvolvimento de soluções específicas para cada um dos sistemas operacionais. Dentro dessa proposta, a linguagem Go apresentou-se

como a melhor alternativa. Inclusive sendo amplamente adotada pela comunidade no desenvolvimento de ferramentas de linha de comando, tal qual uma das mais populares da atualidade a `kubectl` (Principal ferramenta de interação com o Kubernetes).

Análogo a `kubectl`, a ideia principal do `occlient` e possibilitar ao usuário a interação com o controlador SDN, sem a necessidade de interação direta com a API. Sendo mais amigável e diminuindo a carga de aprendizado para interação com o controlador, removendo a necessidade de apontamento de rotas da API e tratamento de erros, pontos que geram retrabalho significativo, para as equipes que optarem por uma abordagem mais programática, que alternativamente necessitarão desenvolver uma solução para interação direta com a API do controlador.

Assim como no `kubectl`, a ideia é que as configurações de rede possam ser executadas de maneira declarativa, renunciando o modelo imperativo, adotado pelas abordagens tradicionais, através da interação direta com a API do controlador. Dessa maneira, estendendo as possibilidades de gerenciamento para um modelo mais automatizado, baseado em práticas como o GitOps (onde a fonte da verdade para oferecer a infraestrutura como código, é um repositório git) além de práticas adotadas na engenharia de software para versionamento de código, possibilitando uma rastreabilidade dos estados da infraestrutura, com base no histórico de commits do repositório.

4. Demonstração de Implementação e Configuração de uma Rede SDN

O cenário de testes escolhido foi uma topologia mista, composta por dispositivos de domínio óptico (*WDM*, através de transponders cassini) e *Ethernet* (PPDP, através de *switches* utilizando o *stratum* como SO). A topologia retrata um ambiente distribuído, organizado seguindo o padrão *spine-and-leaf*, e utiliza a distribuição *2.5.9.ORANBR* do *ONOS* como controlador, customizada pelo grupo de trabalho para atender aos múltiplos domínios tecnológicos, utilizando o mesmo controlador.

A implementação foi realizado sobre uma VM, com *docker* instalado, e os dispositivos emulados com o *CNETLAB*. Após os dispositivos terem sido provisionados, foram enviadas as informações, declarando as responsabilidades de cada um dos dispositivos para o *ONOS*, através do *OCClient*.

A abordagem pretendida é que seja possível replicar toda as configuração de rede através de um repositório *git*, corroborando, inclusive, para o registro de mudanças da topologia, além de abrir espaço para a confecção de automações que possam operar sobre esses arquivos, cujas configurações da topologia estão declaradas. Nesse cenário o *OCClient* aparece como uma solução para as interações básicas na API do *ONOS*, sem que haja a necessidade de desenvolvimento.

4.1. Abordagem *GitOps*

Durante o desenvolvimento da ferramenta, notou-se um potencial para a utilização de práticas de entrega continua de software, através de *pipelines*, com base na abordagem *GitOps*. E dada a característica de distribuição através de binário do *OCClient*, poderíamos utilizar nas mais diversas plataformas de *pipelines*. A seguir, explicamos um pouco sobre *GitOps* e destacamos algumas ferramentas de *pipelines* onde o *OCClient* poderia ser introduzido.

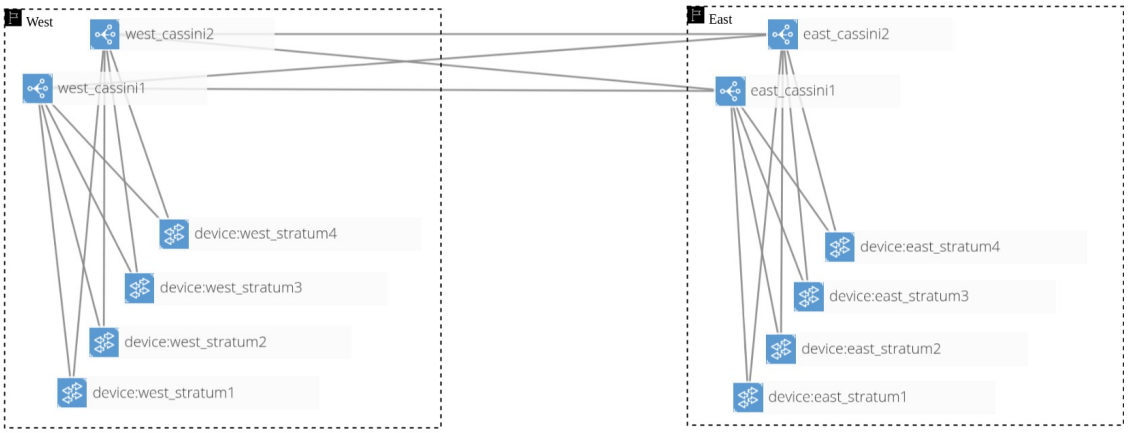

Figura 1. Cenário utilizado para os testes

Utiliza um repositório *git* como fonte para entrega de infraestrutura e gerenciamento de configurações, normalmente estão atrelados a uma plataforma de *Continuous Integration (CI)*, para realizar as ações primárias, como avaliações de sintaxe, formatação e políticas de boas práticas definidas. Além disso, uma plataforma de *Continuous Delivery (CD)*, deve compor a pilha para realizar efetivamente a entrega, aplicando as configurações declaradas no repositório nos dispositivos. Ambas ações podem ser executadas na mesma ferramenta, mas isso dependerá exclusivamente da pilha de ferramentas adotadas.

Alguns exemplos de ferramentas são: *gitlab-ci*, *jenkins* e *github actions*, cuja finalidade é principalmente a integração continua (CI), mas também podem ser utilizadas para realização das entregas. Por outro lado, temos o *ArgoCD*, plataforma de CD, desenvolvida exclusivamente para a realização de entrega, e amplamente adotada no mercado para entregas de infraestrutura baseada em *GitOps*.

Ao utilizar uma abordagem *GitOps* para a entrega de infraestrutura, deixamos de ter um cenário cuja configuração dos dispositivos requer ação imperativa, como e execução manual de um *script* que auxilie o usuário na no ingresso de um dispositivo na topologia, para um cenário onde um dispositivo é declarado, fazendo com que o trabalho que seria executado manualmente possa ser automatizado através das plataformas de integração e entrega continua, como as citadas anteriormente.

Atualmente o principal cenário de uso do *GitOps* é a entrega de aplicações no *Kubernetes*, uma plataforma para execução de sistemas distribuídos em *containers*, ou seja, com alta complexidade de administração. Tendo em vista a complexidade que um *cluster* de controladores ONOS pode ter, emerge-se a ideia de utilizar essa estratégia para entregar as configurações dos dispositivos utilizando um metodologia similar, possibilitando inclusive uma abordagem ao gerenciamento baseada em estados, ao utilizar as versões do repositório.

5. Resultados

Nesse momento, o objetivo com a ferramenta é realizar as operações primárias de inserção, listagem e remoção dos principais componentes (*devices*, *links* e *apps*) do controlador, sem a necessidade de interagir com o controlador de outra forma, como a interface

web, ou a CLI do controlador, comprovando a eficácia da ferramenta para os casos primários, podemos expandir para casos de uso mais específicos e complexos no futuro. Assim como acontece com o `kubectl` ao declara um *deployment*, abstraindo os demais componentes, como o *replicaset*, que precisam ser criados para que o resultado final esperado, a execução de um ou mais *pods*, possa acontecer.

Tomando como base a ISO/IEC 9126, os principais resultados são relacionados a: Funcionalidade, atendendo as requisitos inicialmente expostos; confiabilidade, potencializada pela utilização em conjunto com uma plataforma de integração e entrega continua; usabilidade, pois segue as boas práticas de desenvolvimento de CLIs do projeto GNU; eficiência, pois é disponibilizado em forma de binário removendo a necessidade de configurações adicionais, potencialmente danosas para a estabilidade do sistemas que executa-a; manutenibilidade, pois é escrita uma linguagem popular em repositório que possibilita o trabalho descentralizado e assíncrono dos contribuintes; e portabilidade, pois o mesmo código pode ser compilado e transformado em binário aceito por outros sistemas operacionais, a partir de qualquer sistema operacional moderno (linux, windows, macos).

5.1. Pipeline de configuração via GitOps

A utilização do OCClient localmente mostrou-se muito prática para a realização de testes em ambientes controlados. Entretanto, outra possibilidade de uso que mostrou-se bastante promissora, foi a execução em ambientes de entrega continua e deploy, como o gitlab-ci, github-actions, jenkins e ArgoCD.

Por ser disponibilizado através de um binário, pode ser facilmente integrada na maioria dos ambientes de integração continua e deploy (CI/CD). Além de poder ser executada na maioria dos ambientes de usuário final, como um administrador da rede. Principalmente nesse cenário, sendo muito mais intuitiva que a utilização direta da API do ONOS.

A utilização em ambientes de CI/CD abre possibilidade para automação no gerenciamento dos dispositivos, não apenas das suas configurações, como é possível por meio de ferramentas de gerenciamento de configuração como o ansible [Ansible 2024], salt [Salt 2024] e puppet. Ademais, dadas as proporções dos ambientes atuais, a utilizações de automações é essencial para mitigar falha humana, haja vista que nos cenários de integração e entrega continua políticas de boas práticas são restritores para entregas de má qualidade, ou que potencialmente possam causar instabilidade no ambiente.

Outra grande vantagem do GitOps, é o versionamento da infraestrutura em estados, que torna possível a recuperação rápida, tanto de forma automática, baseada em regras definidas pelos administradores da rede, quanto de forma manual, para qualquer estado registrado no repositório. Dessa forma, aumentando drasticamente a recuperabilidade da rede e a confiabilidade dos sistemas subjacentes atendidos por ela.

5.2. Comparação entre as Soluções de Gerenciamento

Em comparação com a solução padrão, interação direta com a API REST do ONOS, a utilização do OCClient mostrou-se muito mais eficiente, dado que a complexidade de interação com a interação direta é substituída por uma interface mais amigável ao usuário, dado-lhe os poderes de poder executar todas as principais operações, ingresso e remoção de dispositivos, links e portas, através de uma linha de comando e arquivos declarativos.

```
root@cnetlab:~/occlient# ./occlient
OCClient (ONOS Classic Client) facilitates interaction with the ONOS
Classic SDN Controller by sending and querying information via the CLI

Usage:
  occlient [command]

Available Commands:
  applications Manage ONOS applications
  completion   Generate the autocompletion script for the specified shell
  devices      Manage ONOS devices
  help         Help about any command
  links        Manage inventory of infrastructure links
  netconfig    Manage network configurations
  optical      Manage network intent configurations

Flags:
      --config string   config file (default is $HOME/.occlient/config)
  -h, --help            help for occlient
  -s, --server string   ONOS address in IP:Port format (default "localhost:8181")
  -v, --version         version for occlient

Use "occlient [command] --help" for more information about a command.
```

Figura 2. Principais funcionalidades de gerenciamento do OCClient

6. Conclusões e Trabalhos Futuros

Ao realizar uma comparar quantitativamente, a carga de conhecimentos necessários para interação através da API *REST/http* e o *OCClient*, nota-se uma diminuição expressiva. O conhecimento de uma linguagem de programação e interação com APIs *REST/http*, são os dois principais pontos, restando apenas a sintaxe do *JSON*, um padrão de formato aberto troca de informações entre sistemas. Além da extinção da necessidade de conhecer as rotas de API do controlador, proporcionando ao utilizador focar apenas no que é necessário, o *payload* adequado para o caso de uso em questão.

Além disso, vale ressalta que existem operações, como a remoção de dispositivos que não podem ser executadas através da interface *web* do ONOS, exigindo que os operadores tivessem o conhecimentos sobre essas tecnologias que são abstraídas pelo uso do *OCClient*.

Com isso, os objetivos propostos, de extinguir a interação direta com a API do controlador nos casos básicos e oferecer uma interface amigável para a execução das operações de gerenciamento no controlador, foi atingido com sucesso. Oferecendo ao utilizador um ambiente para a realização dessas interações, exigindo o mínimo de conhecimento e proporcionando o mesmo resultado.

Atualmente a interação ainda se baseia nas rotas da API do controlador. Entretanto, em versões futuras busca-se diminuir ainda mais a carga de conhecimentos necessários para a interação com o controlador, através da interação entre a ferramenta e os dispositivos gerenciados, a fim expandir as formas de interação para a execução de fluxos mais complexos, que a adição, listagem de remoção de dispositivos.

Além disso, a integração com controladores de outros domínios tecnológicos, tal como o µONOS, voltado para o domínio de RAN, destaca-se. Principalmente por esses

domínios tecnológicos utilizarem em seus controladores a arquitetura gRPC/http, menos comum atualmente, que geraria um custo ainda maior tanto de tempo quanto de pessoal.

Agradecimentos

Este trabalho foi realizado com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), da Rede Nacional de Ensino e Pesquisa (RNP) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), por intermédio dos projetos No. 2023/00811-0, No. 2020/04031-1, No. 2021/00199-8, e projeto No. 2018/23097-3.

Referências

Ansible (2024). Ansible. Disponível em: `https://github.com/ansible/ansible`. Acesso em: 10 mar. 2024.

Liatifis, A., Sarigiannidis, P., Argyriou, V., and Lagkas, T. (2023). Advancing sdn from openflow to p4: A survey. *ACM Comput. Surv.*, 55(9).

ONF (2023). Open network operating system (onos). Disponível em: `https://opennetworking.org/onos/`. Acesso em: 29 mar. 2024.

RNP (2023). Cnetlab - networking digital twin environment. Disponível em: `https://git.rnp.br/cnar/sdn-multicamada/emulacao/emulador-optico`. Acesso em: 12 fev. 2024.

RNP, C. et al. (2021). Openran brasil. Disponível em: `https://www.rnp.br/projetos/openranbrasil`. Acesso em: 29 mar. 2024.

Salt (2024). Salt. Disponível em: `https://saltproject.io/`. Acesso em: 10 mar. 2024.

Silva, M. et al. (2023a). Oran-onos. Disponível em: `https://hub.docker.com/r/muriloavlis/oran-onos`. Acesso em: 13 fev. 2024.

Silva, M., Gomes, M., Dias, V., Oliveira, L., Farias, F., and Abelém, A. (2023b). Redes definidas por software para a orquestração de diferentes domínios tecnológicos. In *Anais do XIV Workshop de Pesquisa Experimental da Internet do Futuro*, pages 19–24, Porto Alegre, RS, Brasil. SBC.