GABRIEL FLUGEL SILVA

AUTOMAÇÃO DE REDE SD-WAN USANDO PYTHON COM MIKROTIK

Ji-Paraná
2023

GABRIEL FLUGEL SILVA

AUTOMAÇÃO DE REDE SD-WAN USANDO PYTHON COM MIKROTIK

Trabalho de Conclusão de Curso apresentado ao Centro Universitário São Lucas Ji-Paraná como requisito parcial para obtenção do Título de Bacharel em Sistemas de Informação. Prof. Orientador: Romário Vitorino Ferreira.

Ji-Paraná

Dados Internacionais de Catalogação na Publicação - CIP

S586a Silva, Gabriel Flugel.

Automação de rede SD-WAN usando Python com Mikrotik. / Gabriel Flugel Silva. – Ji-Paraná, 2023.
27 p.; il.

Trabalho de Conclusão de Curso (Sistemas de Informação) – Centro Universitário São Lucas Ji-Paraná, 2023.

Orientador: Prof. Esp. Romário Vitorino Ferreira.

1. Python. 2. Mikrotik. 3. SD-WAN. 4. Internet. 5. Protocolo. I. Ferreira, Romário Vitorino. II. Título.

CDU 681.51

Ficha Catalográfica Elaborada pelo Bibliotecário Giordani Nunes da Silva CRB 11/1125

AGRADECIMENTOS

Quero expressar minha profunda gratidão a todos que contribuíram para a conclusão deste Trabalho de Conclusão de Curso em Sistemas de Informação.

Em primeiro lugar, agradeço a meu orientador, Romário Vitorine, pela sua orientação dedicada e valiosos insights ao longo deste processo. Sua orientação foi fundamental para a qualidade deste trabalho.

Agradeço também aos professores do curso de Sistemas de Informação, que compartilharam seus conhecimentos e experiências, proporcionando uma base sólida para o desenvolvimento deste trabalho.

À minha família e amigos, agradeço pelo constante apoio e incentivo. Suas palavras de encorajamento foram essenciais para superar os desafios encontrados durante essa jornada acadêmica

SUMÁRIO

RESUMO

Nesse contexto, a tecnologia Software-Defined Wide Area Network (SD-WAN) tem se destacado como uma abordagem inovadora para otimizar o desempenho e a eficiência das redes de computadores. De SD-WAN (Rede de Área Ampla Definida por Software) é uma arquitetura de rede que utiliza a tecnologia de virtualização e o software para proporcionar uma abordagem mais dinâmica e com maior agilidade para a gestão de redes de ampla área. "Através de uma arquitetura computacional estruturada por hardware e software, ela consegue virtualizar todas as conexões WAN e oferecer diferentes serviços, de acordo com as necessidades das organizações." (Durbano, 2019). Ao contrário das redes tradicionais baseadas em hardware, a SD-WAN permite a centralização do controle, a segmentação do tráfego e a implementação de políticas de acordo com as necessidades específicas da organização. Conforme aponta Durbano (2019), isso resulta em uma melhor utilização dos recursos de rede disponíveis, além de permitir uma gestão simplificada e uma visibilidade aprimorada sobre o tráfego e o desempenho da rede

Palavras-chave: Python, Mikrotik, SD-WAN, Internet, Protocolo

ABSTRACT

In the contemporary landscape of computer networks, the innovative approach of Software-Defined Wide Area Network (SD-WAN) has emerged as a prominent solution to optimize performance and efficiency. SD-WAN leverages a Software-Defined Networking (SDN) architecture, utilizing virtualization and software to offer a more flexible and agile approach to the management of wide area networks. Through a hardware and software-structured computational architecture, SD-WAN effectively virtualizes all Wide Area Network (WAN) connections, providing diverse services tailored to the specific needs of organizations (Durbano, 2019). Diverging from traditional hardware-based networks, SD-WAN enables centralized control, traffic segmentation, and the implementation of policies customized to the unique requirements of each organization. As highlighted by Durbano (2019), this approach leads to enhanced utilization of available network resources, facilitating simplified management, and providing improved visibility into network traffic and performance.

Keywords: Python, Mikrotik, SD-WAN, Internet, Network Protocol

1 INTRODUÇÃO

As redes de computadores desempenham um papel crucial na comunicação e troca de informações nas organizações contemporâneas (BEAL, 2009). Diante do crescente requisito por conectividade, da maior dependência de aplicativos baseados em nuvem e da necessidade de assegurar uma experiência de usuário de alta qualidade, surgem desafios substanciais para as redes de computadores, abrangendo áreas como desempenho, segurança e gerenciamento eficaz.

Nesse cenário, destaca-se a tecnologia SD-WAN como uma abordagem inovadora para otimizar o desempenho e a eficiência das redes de computadores. Segundo Durbano (2019), a SD-WAN (Rede de Área Ampla Definida por Software) é uma arquitetura de rede que emprega virtualização e software, proporcionando uma abordagem mais ágil e flexível para a gestão de redes extensas.

Através de uma estrutura computacional integrada por hardware e software, a SD-WAN consegue virtualizar todas as conexões WAN, oferecendo uma gama diversificada de serviços adaptados às necessidades específicas das organizações (Durbano, 2019). Em contraste com as redes tradicionais baseadas em hardware, a SD-WAN possibilita a centralização do controle, a segmentação do tráfego e a implementação de políticas alinhadas às particularidades da organização. Conforme salientado por Durbano (2019), esse enfoque resulta em uma otimização mais eficiente dos recursos de rede disponíveis, promovendo uma gestão simplificada e uma visibilidade aprimorada sobre o tráfego e o desempenho da rede.

2. MATERIAL E MÉTODOS

Nesta seção serão apresentados os materiais e métodos do trabalho.

2.1. Ambiente

O ambiente no qual as operações foram executadas e os resultados foram apresentados foi o sistema operacional Ubuntu Server. Essa escolha proporciona uma base sólida e confiável para a implementação do sistema SD-WAN, assegurando uma execução eficiente e consistente das operações propostas. O Ubuntu Server oferece um ambiente robusto que contribui para a estabilidade e desempenho do sistema, garantindo uma experiência confiável durante toda a validação e análise dos resultados.

3.2. Equipamento

Utilizou-se o equipamento da Mikrotik pois seu custo é baixo e entregua muitas ferramentas para rede. O modelo que utilizados no teste foi RB3011UiAS-RM. De acordo com a propria fabricante, é dispositivo multiportas, o primeiro a executar uma CPU de arquitetura ARM para um desempenho mais alto do que nunca. O RB3011 possui dez portas Gigabit divididas em dois grupos de switches, um compartimento SFP e pela primeira vez uma porta SuperSpeed full size USB 3.0, para adicionar armazenamento ou um modem 3G/4G externo.

Especificações técnica:

- **Arquitetura**: ARM 32 bits
- **CPU**: IPQ-8064
- **Contagem de núcleos da CPU**: 2
- **Frequência nominal da CPU**: 1,4GHz
- **Modelo de chip de comutação**: QCA8337
- **Dimensões**: 443x92x44mm
- **Licença RouterOS**: 5 • **Sistema operacional**: RouterOS • **Tamanho da RAM**: 1 GB • **Tamanho de armazenamento**: 128MB
- **Tipo de armazenamento**: NAND
- **MTBF**: Aproximadamente 200.000 horas a 25C
-

3.3 Linguagem Python

Ao decidir a linguagem de programação para a implementação de soluções de otimização de redes, Python se destaca como uma escolha poderosa e versátil. Sua sintaxe clara, extensa biblioteca padrão e comunidade ativa tornam Python uma ferramenta ideal para desenvolver aplicações eficientes e de fácil manutenção. A utilização da linguagem Python ganha ainda mais relevância ao considerar a existência da API "ROS_API" para a comunicação direta com os equipamentos MikroTik. Essa API proporciona uma interface eficaz para interagir com os dispositivos, permitindo a manipulação de configurações e o gerenciamento de rotas de forma simplificada. A simplicidade e expressividade do Python se alinham perfeitamente com a filosofia de desenvolvimento eficiente, fundamental ao lidar com a complexidade inerente à otimização de redes. A capacidade de integrar facilmente a API "ROS_API" ao código Python oferece uma solução elegante e robusta para a comunicação com os equipamentos MikroTik, simplificando o processo de implementação e manutenção do sistema. Além disso, a vasta comunidade Python e a abundância de recursos online facilitam o aprendizado contínuo e a resolução de problemas, garantindo um ambiente propício ao desenvolvimento ágil e inovação na otimização de redes.

3.4. Instalação da linguagem Python

O Python é uma linguagem de programação versátil e amplamente utilizada, e instalar a versão mais recente no seu sistema Ubuntu Server proporcionará um ambiente de desenvolvimento robusto.

3.4.1 Atualizar o sistema

Antes de começar, é sempre uma boa prática atualizar os pacotes do sistema. Execute os seguintes comandos no terminal:

- Sudo apt updat
- Sudo apt upgrade -y

3.4.2 Verificação da Versão do Python

O Ubuntu geralmente vem com o Python pré-instalado. Para verificar a versão instalada, use o comando:

- python3 –version

3.4.3. Instalação do Python 3

Se o Python não estiver instalado ou se você quiser instalar a versão mais recente, utilize o seguinte comando:

- sudo apt install python3
-

3.4.4. Verificação da Instalação

Após a conclusão da instalação, verifique novamente a versão do Python:

- python3 –version

3.4.5 Instalação do pip (Gestor de Pacotes do Python)

O pip é utilizado para instalar bibliotecas e pacotes do Python. Instale-o com o seguinte comando:

- sudo apt install python3-pip

3.4.6 Verificação do pip

Confirme a instalação do pip verificando sua versão:

- pip3 –version

Agora tem-se o Python e o pip instalados no seu Ubuntu Server. Este ambiente fornecerá uma base sólida para o desenvolvimento de aplicativos e projetos Python. Certifique-se de ajustar os comandos conforme necessário, dependendo da versão específica do Python que você deseja instalar ou se precisar de permissões de superusuário.

3.5. Instalação do git

Para instalar o sistema que nomeamos de PY-WAN, o acesso está em nosso github (https://github.com/Flugelo/pywan-tcc.git). Para executar a instalação, deve-se seguir os seguintes comandos:

3.5.1 Instalação do git no sistema operacional ubuntu

Execute o seguinte comando para instalação do git

- sudo apt install git

3.5.2 Verificação da instalação

Confirme se a instalação foi bem-sucedida verificando a versão do Git:

- git –version

3.5.3 Configurações iniciais do git

Configure seu nome de usuário e endereço de e-mail para associar às suas contribuições:

- git config --global user.name "Seu Nome"
- git config --global user.email seu@email.com

Substitua "Seu Nome" e "seu@email.com" com suas informações.

Agora, o Git está instalado e configurado no seu Ubuntu Server. Você está pronto para começar a usar o Git para rastrear e gerenciar versões dos seus projetos.

3.5.4 Instalação do algoritmo do projeto

Para executarmos os próximos passo, devemos instalar o algoritmo que foi desenvolvido para aplicação da SD-WAN, que chamamos de PY-WAN.

3.5.5 Clone do repositório do projeto

Para clonarmos o repositório onde se encontra o projeto, executamos o comando:

• git clone https://github.com/Flugelo/pywan-tcc.git

Em seguida, navegue até o diretório recém-clonado: **cd pywan-tcc**

3.5.6 Criar e Ativar o ambiente virtual Python

Dentro do diretório do projeto, crie um ambiente virtual do Python:

- python -m venv venv Ative o ambiente virtual:
- source venv/bin/activate

3.5.6. Instalar Bibliotecas Necessárias

Instale as bibliotecas necessárias listadas no arquivo requirements.txt:

- pip install -r requirements.txt

3.5.7. Configurar o arquivo de Ambiente

Habilite o arquivo de configuração do projeto:

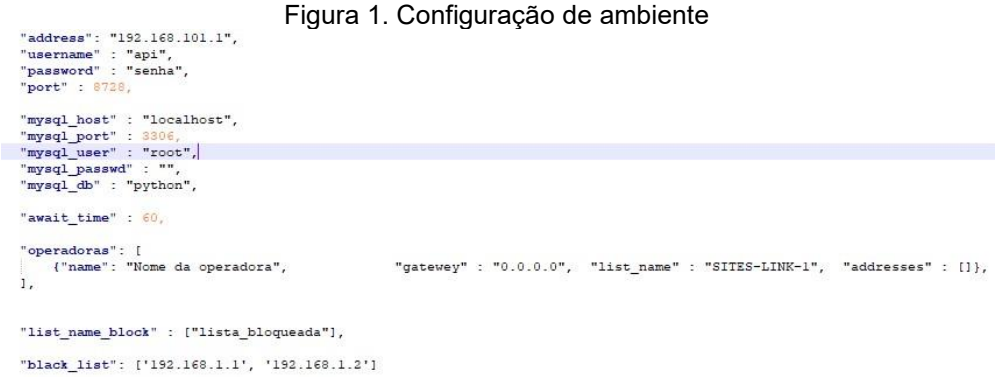
- mv config.copy.env config.json

Acesse o arquivo de configuração para personalizar as seguintes informações:

- nano config.json
- Configuração de acesso ao banco de dados para armazenamento dos testes e manipulações.
- Configuração de acesso à API do equipamento MikroTik.
- Configurações das operadoras, incluindo IP gateway, nome e a lista responsável pela operadora.

  Na Figura 1 apresentada abaixa tem um exemplo desta configuração.

Figura 1. Configuração de ambiente



```
"address": "192.168.101.1",
"username" : "api",
"password" : "senha",
"port" : 8728,

"mysql_host" : "localhost",
"mysql_port" : 3306,
"mysql_user" : "root",
"mysql_passwd" : "",
"mysql_db" : "python",

"await_time" : 60,

"operadoras": [
    {"name": "Nome da operadora",            "gatewey" : "0.0.0.0",  "list_name" : "SITES-LINK-1",  "addresses" : []},
],


"list_name_block" : ["lista_bloqueada"],

"black_list": ['192.168.1.1', '192.168.1.2']
```

Fonte: Autor, 2023

OBS: Pode-se também configurar uma lista de listas de endereços ip caso você tenha configurado em seu equipamento. Podemos também aplicar uma black list em endereços ip especificos.

3.6. Execução do sistema

Nesta seção serão apresentadas as etapas para execução do sistema.

3.6.1 Verificação dos endereços ip no endereço de lista

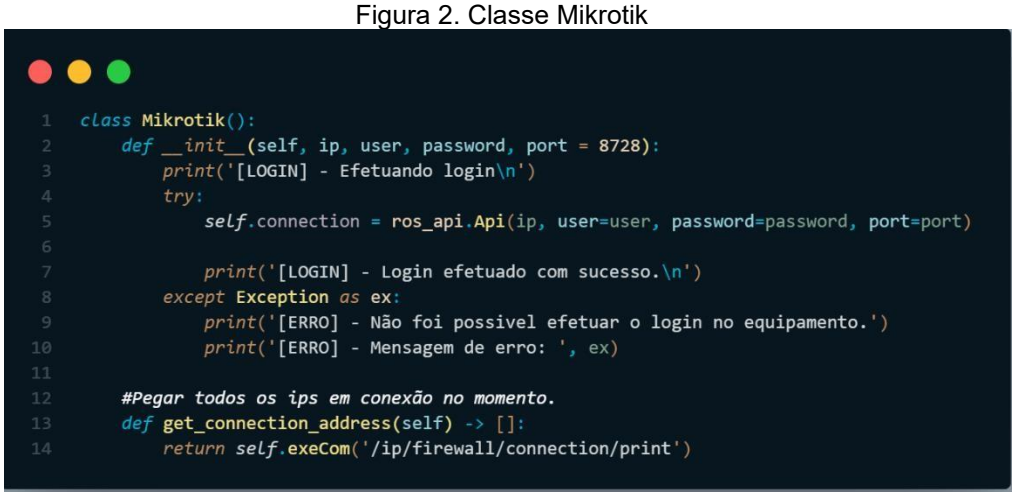
Com a função get_address_list() dentro da classe mikrotik, o algoritmo coleta todos os endereços ip já manipulados em “'/ip/firewall/address-list/print”.

- Verifica endereço por endereço.
- Normaliza o endereço ip coletado na address-list.
- Verifica se a normalização retornou vazio.
- Valida se o endereço ip não é privado.
- Valida se o endereço coletado na address-list não está na black list.
- Valida se a lista do endereço ip coletado na address-list não esta também em black list.
- Transforma o endereço ip em um bloco ip /24.
- Valida se a transformação do bloco não retornou vazia.
- Insere os endereços ip com sua lista em um array de objetos do tipo NetworkAddress.

3.6.2 Coleta de endereços IP do equipamento

O algoritmo coleta os dados executando o seguinte o método get_connections_address() que se encontra na classe Mikrotik, responsável pela autenticação e comandos executados. O método get_connections_address() executa o seguinte comando “ip/firewall/connection/print”. Conforme apresentado na Figura 2 abaixo.

Figura 2. Classe Mikrotik



```python
class Mikrotik():
    def __init__(self, ip, user, password, port = 8728):
        print('[LOGIN] - Efetuando login\n')
        try:
            self.connection = ros_api.Api(ip, user=user, password=password, port=port)

            print('[LOGIN] - Login efetuado com sucesso.\n')
        except Exception as ex:
            print('[ERRO] - Não foi possivel efetuar o login no equipamento.')
            print('[ERRO] - Mensagem de erro: ', ex)

    #Pegar todos os ips em conexão no momento.
    def get_connection_address(self) -> []:
        return self.exeCom('/ip/firewall/connection/print')
```

Fonte: Autor, 2023

3.6.3. Validação dos Endereços Ips

Depois das coletas dos endereços, o algoritmo executara as seguintes validações:

- Normalizar o endereço ip retornado do comando “ip/firewall/connection/print”.
- Verificação se o endereço normalizado não é vazio. Caso seja vazio o código apenas pula o endereço ip.
- Com a função is_private_ip() dentro da classe tools, verifica se o endereço ip normalizado não é um endereço de ip privado. Caso seja um endereço ip privado o código apenas pula o endereço ip.
- Verifica se o endereço ip está na black list. Caso esteja na black list o código apenas pula o endereço ip.
- Transforma o endereço ip em um bloco de endereço ip /24. Caso seja vazio o código apenas pula o endereço ip.
- Verifica se a transformação retornou um valor vazio. Caso seja vazio o código apenas pula o endereço ip.
- Verifica os endereços ip já coletados da address-list anterior.
- Verifica se nos blocos de endereços ip /24 já existe.

- Caso exista, o sistema adiciona o ip dentro do bloco /24
- Caso não exista, o sistema cria um bloco ip /24 com o endereço ip testado.

Na Figura 3 apresentada abaixo é possível verificar o trecho de código para este cenário.

Figura 3. Verifica endereço por endereço

```python
#Verificar endereço por enderço
    for conn in connections:

        #Normalizar o endereço IP
        address_cleared = conn['dst-address'].split(':')[0]

        #Verificar se a normalização não retornou um endereço vazio
        if address_cleared is None: continue

        #Verifica se o endereço não é um ip privado.
        if tools.is_private_ip(address_cleared): continue

        #Verifica o endereço ip não esta na black list;
        if address_cleared in black_list: continue


        #Transforma o endereço em um bloco /24;
        address_with_mask = tools.get_block_ip(address_cleared, '24')

        #Verifica se a transformação do endereço ip para um bloco de endereço ip não é vazio.
        if address_with_mask is None: continue
```

```python
in_inside = False
    for block in list_block_address:
        if tools.inside_block_address(block=block.network, address=address_cleared):

            if address_cleared not in block.addresses:
                block.addresses.append(address_cleared)

            if block.address == None: block.address = address_cleared

            in_inside = True
            break

    if not in_inside:
        list_block_address.append(NetworkAddress(network=address_with_mask, address=address_cleared, addresses=[address_cleared]))
```

Fonte: Autor, 2023

3.4 Testes de Latência entre Operadoras Cadastradas

Depois das validações no endereço ip, o sistema começara a executar os teste de latência pela operadoras configuradas no arquivo config.json.

- Valida se o endereço ip que será testado não é um valor vazio.
- Começa a executar o teste de latência entre as operadoras.

Na Figura 4 presentada abaixo é possível verificar o trecho de código para este cenário

Figura 4. Executando Ping com IP destino e Origem

```
#Executar um ping com ip de destino e origem.
    def ping(self, address : str, src: str) -> []:
        return self.exeCom('/ping \n =address=' + address +' \n =src-address=' + str(src) +' =count=1')
```

Fonte: Autor, 2023

Executa a função ping() da classe mikrotik que executa o comando.Na Figura 3 apresentada abaixo é possível verificar o trecho de código para este cenário.

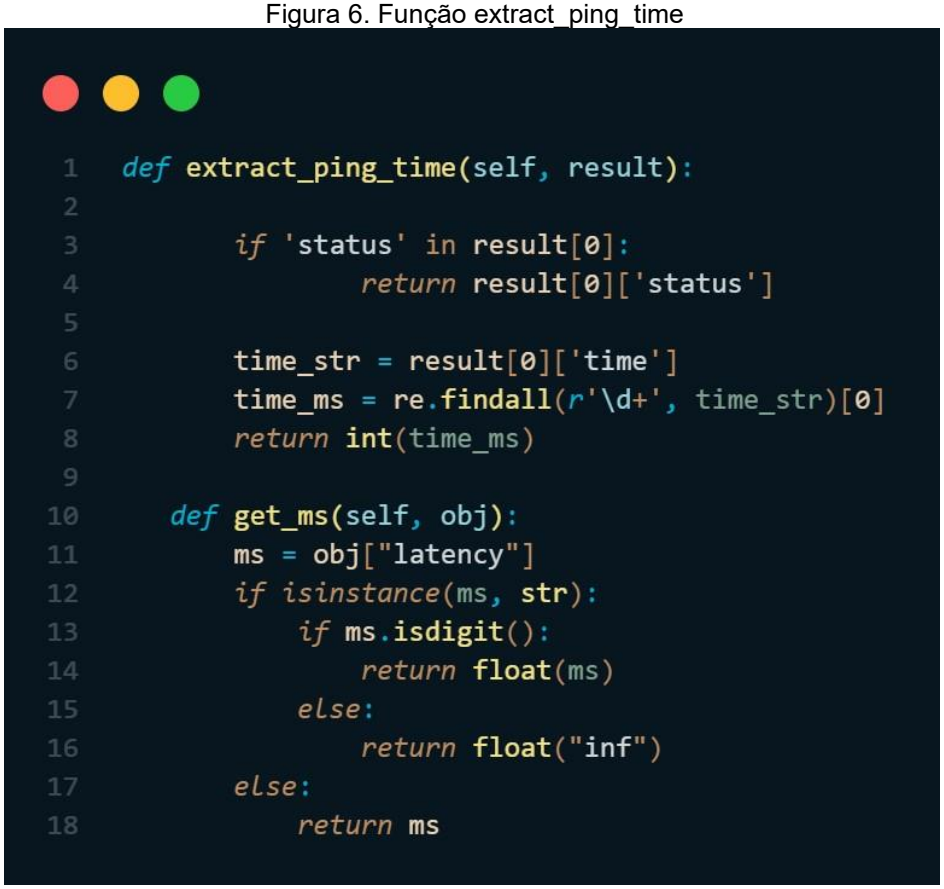
Figura 5. Função Ping

```
def ping(self, address : str, src: str) -> []:
        return self.exeCom('/ping \n =address=' + address +' \n =src-address=' + str(src) +' =count=1')
```

Fonte: Autor, 2023

Normaliza a resposta do comando em um valor inteiro chamando a função extract_ping_time() dentro da classe tools. Na Figura 6 apresentada abaixo é possível verificar o trecho de código para este cenário.

Figura 6. Função extract_ping_time

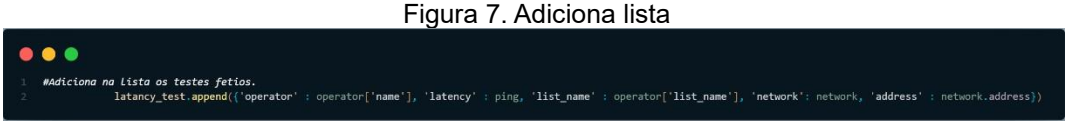


```
def extract_ping_time(self, result):

        if 'status' in result[0]:
                return result[0]['status']

        time_str = result[0]['time']
        time_ms = re.findall(r'\d+', time_str)[0]
        return int(time_ms)

    def get_ms(self, obj):
        ms = obj["latency"]
        if isinstance(ms, str):
            if ms.isdigit():
                return float(ms)
            else:
                return float("inf")
        else:
            return ms
```

Fonte: Autor, 2023

Adiciona o teste feito em uma lista. Na Figura 7 apresentada abaixo é possível verificar o trecho de código para este cenário.

Figura 7. Adiciona lista



Fonte: Autor, 2023

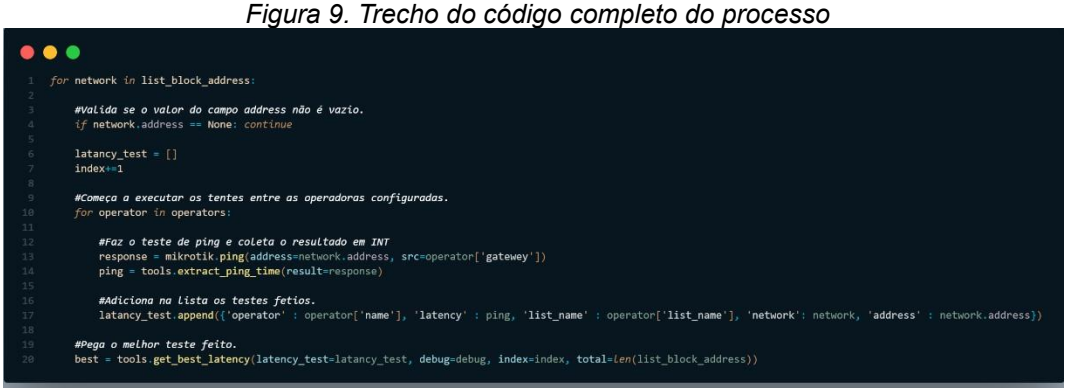
Valida o melhor teste com a menor latência chamando o método get_best_latency() da classe tools. Na Figura 8 apresentada abaixo é possível verificar o trecho de código para este cenário.

Figura 8. Função Get_best_latency

Fonte: Autor, 2023

Na Figura 9 apresentada abaixo é possível verificar o trecho de código completo para este processo.

Figura 9. Trecho do código completo do processo

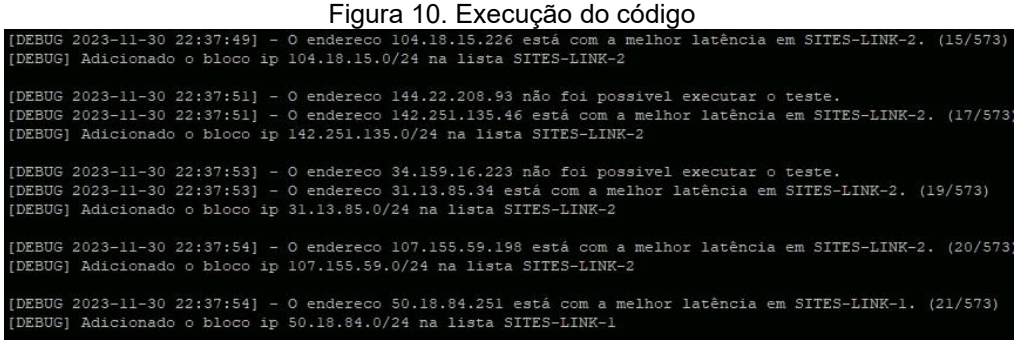


Fonte: Autor, 2023

3.6.3. Execução do codigo

Após a validação, o sistema comunica, por meio da Interface de Linha de Comando (CLI), as ações que está realizando, abrangendo desde a execução de testes de ping até a otimização da latência, assim como a remoção e adição de novas rotas conforme necessário. Na Figura 3 apresentada abaixo é possível verificar o o resultado após a execução do código.

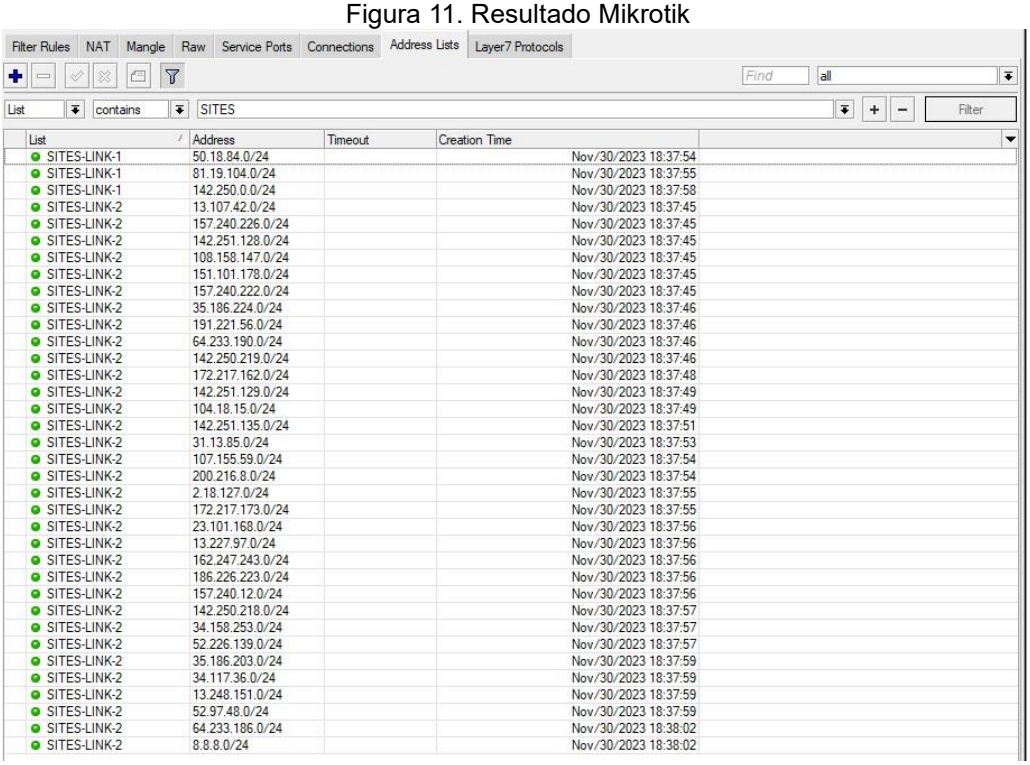
Figura 10. Execução do código



Fonte: Autor, 2023

As informações mostradas na imagem acima, pode ser vista também no proprio equipamento da mikrotik, na aba IP -> FIREWALL. Na Figura 11 apresentada abaixo é possível verificar as informações apresentadas no Mikrotik.

Figura 11. Resultado Mikrotik



List	Address	Timeout	Creation Time
SITES-LINK-1	50.18.84.0/24		Nov/30/2023 18:37:54
SITES-LINK-1	81.19.104.0/24		Nov/30/2023 18:37:55
SITES-LINK-1	142.250.0.0/24		Nov/30/2023 18:37:58
SITES-LINK-2	13.107.42.0/24		Nov/30/2023 18:37:45
SITES-LINK-2	157.240.226.0/24		Nov/30/2023 18:37:45
SITES-LINK-2	142.251.128.0/24		Nov/30/2023 18:37:45
SITES-LINK-2	108.158.147.0/24		Nov/30/2023 18:37:45
SITES-LINK-2	151.101.178.0/24		Nov/30/2023 18:37:45
SITES-LINK-2	157.240.222.0/24		Nov/30/2023 18:37:45
SITES-LINK-2	35.186.224.0/24		Nov/30/2023 18:37:46
SITES-LINK-2	191.221.56.0/24		Nov/30/2023 18:37:46
SITES-LINK-2	64.233.190.0/24		Nov/30/2023 18:37:46
SITES-LINK-2	142.250.219.0/24		Nov/30/2023 18:37:46
SITES-LINK-2	172.217.162.0/24		Nov/30/2023 18:37:48
SITES-LINK-2	142.251.129.0/24		Nov/30/2023 18:37:49
SITES-LINK-2	104.18.15.0/24		Nov/30/2023 18:37:49
SITES-LINK-2	142.251.135.0/24		Nov/30/2023 18:37:51
SITES-LINK-2	31.13.85.0/24		Nov/30/2023 18:37:53
SITES-LINK-2	107.155.59.0/24		Nov/30/2023 18:37:54
SITES-LINK-2	200.216.8.0/24		Nov/30/2023 18:37:54
SITES-LINK-2	2.18.127.0/24		Nov/30/2023 18:37:55
SITES-LINK-2	172.217.173.0/24		Nov/30/2023 18:37:55
SITES-LINK-2	23.101.168.0/24		Nov/30/2023 18:37:56
SITES-LINK-2	13.227.97.0/24		Nov/30/2023 18:37:56
SITES-LINK-2	162.247.243.0/24		Nov/30/2023 18:37:56
SITES-LINK-2	186.226.223.0/24		Nov/30/2023 18:37:56
SITES-LINK-2	157.240.12.0/24		Nov/30/2023 18:37:56
SITES-LINK-2	142.250.218.0/24		Nov/30/2023 18:37:57
SITES-LINK-2	34.158.253.0/24		Nov/30/2023 18:37:57
SITES-LINK-2	52.226.139.0/24		Nov/30/2023 18:37:57
SITES-LINK-2	35.186.203.0/24		Nov/30/2023 18:37:59
SITES-LINK-2	34.117.36.0/24		Nov/30/2023 18:37:59
SITES-LINK-2	13.248.151.0/24		Nov/30/2023 18:37:59
SITES-LINK-2	52.97.48.0/24		Nov/30/2023 18:37:59
SITES-LINK-2	64.233.186.0/24		Nov/30/2023 18:38:02
SITES-LINK-2	8.8.8.0/24		Nov/30/2023 18:38:02

Fonte: Autor, 2023

Como o algoritmo também armazena os testes executas, temos historico de latência nos endereços ip de destino. Podemos executar relatórios como na imagem abaixo:

Executando o seguinte comando no banco de dados MYSQL

“select list_name, min(latency) as min, avg(latency) as med, max(latency) as max FROM ping WHERE address = '8.8.8.8' GROUP BY list_name ORDER BY med”

Na Figura 12 apresentada abaixo é possível verificar o trecho de código para resultado da consulta ao banco de dados.

Figura 12. Resultado consulta banco de dados

```
+--------------+------+----------+------+
| list_name    | min  | med      | max  |
+--------------+------+----------+------+
| SITES-LINK-6 |   33 |  37.5237 |   42 |
| SITES-LINK-2 |   35 |  37.7784 |   62 |
| SITES-LINK-7 |   28 |  37.8322 |  667 |
| SITES-LINK-1 |   39 |  54.1318 |   72 |
| SITES-LINK-5 |   49 |  61.5000 |   69 |
| SITES-LINK-4 |   29 |  81.9750 |  104 |
| SITES-LINK-3 |   39 | 615.4864 |  981 |
+--------------+------+----------+------+
7 rows in set (0.225 sec)
```

Fonte: Autor, 2023

Resultado os testes de latencia executado para o destino 8.8.8.8, a Google. Observamos que a média da operadora SITES-LINK-6 está melhor que as demais.

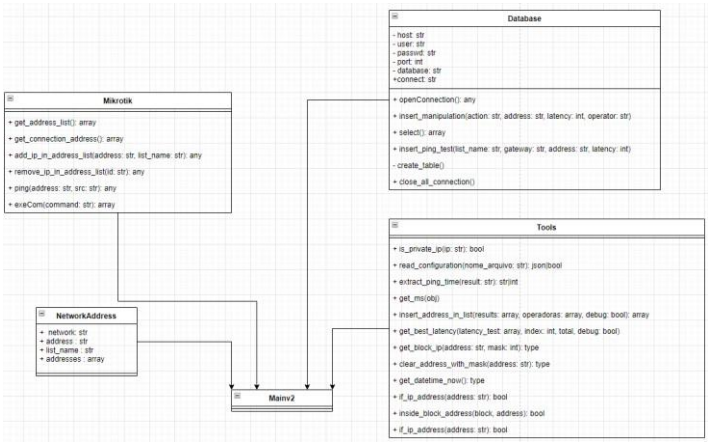
4 MODELAGEM

Nesta seção serão apresentadas as modelagens do software desenvolvido.

4.1 Diagrama de classes

Na Figura 13 abaixo é apresentado o Diagrama de classes do software.

Figura 13. Diagrama de classes

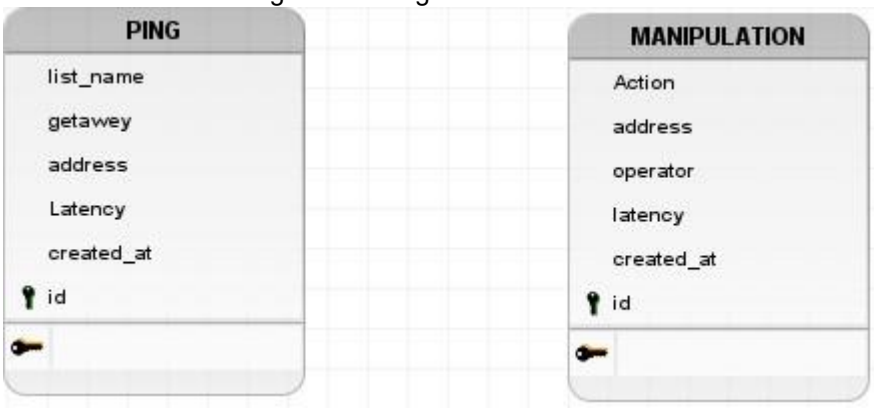


Fonte: Autor, 2023

4.2 Diagrama de entidade

Na Figura 14 abaixo é apresentado o Diagrama de entidade do software.

Figura 14. Diagrama de entidade



Fonte: Autor, 2023

5 RESULTADOS E DISCUSSÃO

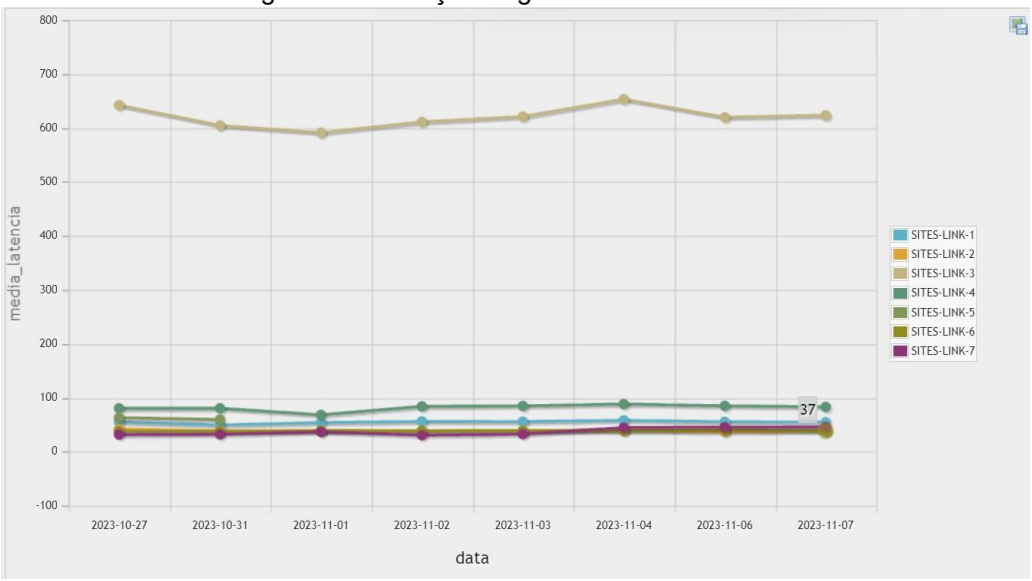
A implementação bem-sucedida do sistema SD-WAN utilizando Python e a API "ROS_API" para comunicação com os equipamentos MikroTik proporcionou resultados notáveis na otimização da rede. Nesta seção, apresentamos e discutimos os principais achados e impactos observados durante a avaliação do sistema.

5.1 Latência Reduzida

Uma análise detalhada revelou uma significativa redução na latência da rede ao implementar o sistema SD-WAN. A utilização eficaz da API "ROS_API" permitiu ajustes dinâmicos nas rotas, direcionando o tráfego para operadoras com menor latência. Isso resultou em tempos de resposta mais rápidos e uma experiência de usuário mais fluida.

Conforme demostrado no grafico abaixo, o sistema coletos os dados diariamente do dentino 8.8.8.8 (google) e fazendo uma média do periodo 27/10/2023 até o dia 07/11/2023, conseguimos vizuliar a estabilidade das operadoras configuradas no sistema. Vejamos que o link numero 6 encontrase com a latência diaria para o google mais estavel. Enquanto os demais link houve alterações significativas. Na Figura 15 abaixo é apresentada o gráfico com as alterações nos links.

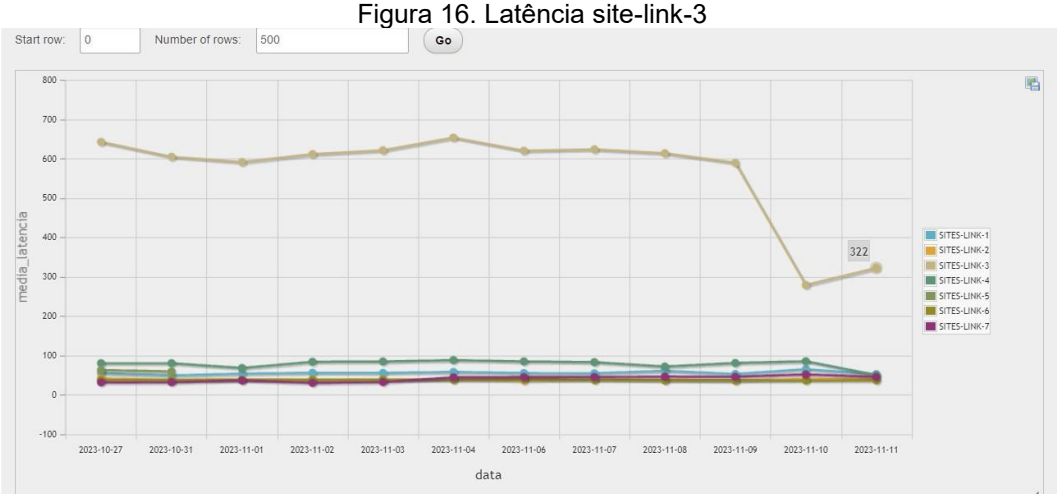
Figura 15. Alterações significativas nos links



Fonte: Autor, 2023

Grafico para o destino 1.1.1.1 Serviço da cloudflare. Notamos que o SITE-LINK-3 sempre operou com a latência muito acima de media de outras operaoras. Na Figura 16 é apresentada a latência acima da média do SITE-LINK-3.

Figura 16. Latência site-link-3



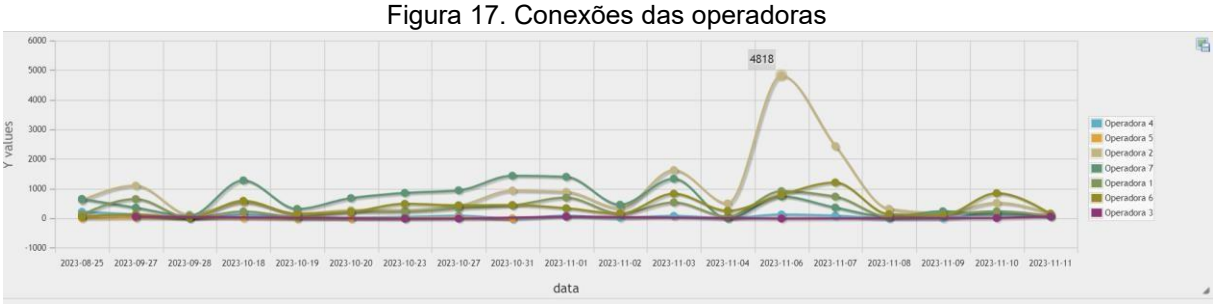
Fonte: Autor, 2023

5.2. Otimização Dinâmica das Rotas

A capacidade de centralizar o controle e segmentar o tráfego conforme as necessidades específicas da organização proporcionou uma otimização dinâmica das rotas. A implementação de políticas personalizadas adaptou-se continuamente às mudanças na demanda de rede, garantindo uma utilização mais eficiente dos recursos disponíveis.

Baseado no grafico abaixo, conseguimos visulizar que no dia 06/11/2023 a operadora 2 teve uma inserção de mais de 4 mil ip, signifcando que nesse dia, as conexões estavam mellhores na operadora 2, conforme apresentada na Figura 17.

Figura 17. Conexões das operadoras



Fonte: Autor, 2023

5.3. Simplificação na Gestão

A centralização do controle conferiu uma gestão simplificada das operações de rede. A integração perfeita entre Python e a API "ROS_API" permitiu a automação de tarefas complexas, reduzindo a carga operacional e facilitando a manutenção contínua.

5.4. Visibilidade Aprimorada

A implementação do sistema proporcionou uma visibilidade aprimorada sobre o tráfego e o desempenho da rede. A combinação de Python e a API "ROS_API" facilitou a coleta de dados precisos, permitindo uma análise mais detalhada e tomada de decisões informadas.

5.5 Discussão

A análise dos resultados destaca não apenas a eficácia da implementação, mas também a adaptabilidade contínua do sistema às demandas dinâmicas da rede. A escolha de Python revelou-se acertada, garantindo não apenas uma implementação eficiente, mas também uma manutenção ágil e escalabilidade futura.

A utilização da API "ROS_API" desempenhou um papel crucial na comunicação eficiente com os equipamentos MikroTik, estabelecendo uma ponte sólida entre o código Python e a infraestrutura de rede. A sinergia entre essas tecnologias reflete-se nos resultados positivos alcançados.

Em síntese, a implementação do sistema SD-WAN com Python e a API "ROS_API" demonstrou um impacto significativo na otimização da rede MikroTik. Os resultados positivos reforçam a viabilidade e eficácia dessa abordagem, consolidando a como uma solução promissora para ambientes que demandam uma gestão dinâmica e eficiente de suas redes.

6 CONCLUSÃO

A presente pesquisa revela a importância crucial das redes de computadores nas comunicações organizacionais modernas, destacando os desafios enfrentados, como demanda crescente por conectividade, dependência de aplicativos em nuvem e a necessidade de garantir experiências de usuário de qualidade. Nesse contexto, a tecnologia SD-WAN surge como uma inovadora abordagem para otimizar o desempenho e a eficiência das redes, utilizando a virtualização e o software para uma gestão mais flexível e ágil, como proposto por Durbano (2019).

Ao implementar o sistema SD-WAN utilizando Python e a API "ROS_API" para comunicação com equipamentos MikroTik, observou-se resultados notáveis. A redução significativa na latência da rede, evidenciada pela rápida resposta e experiência de usuário mais fluida, destaca o impacto positivo da implementação. A otimização dinâmica das rotas, adaptando-se às demandas variáveis, e a simplificação na gestão, com automação de tarefas complexas, reforçam a eficácia do sistema.

A visibilidade aprimorada sobre o tráfego e desempenho da rede proporcionou uma análise mais detalhada e a tomada de decisões informadas. A adaptabilidade contínua do sistema às demandas dinâmicas ressalta sua eficiência e a escolha acertada de Python, não apenas para a implementação eficiente, mas também para a manutenção ágil e escalabilidade futura.

A sinergia entre Python e a API "ROS_API" desempenhou um papel crucial, estabelecendo uma ponte sólida entre o código e a infraestrutura de rede. Em síntese, a pesquisa confirma que a implementação do sistema SD-WAN é uma solução promissora para ambientes que buscam uma gestão dinâmica e eficiente de suas redes, oferecendo resultados positivos e consolidando a relevância dessa abordagem na otimização de redes MikroTik.

7. REFERÊNCIAS BIBLIOGRÁFICAS

Explicação sobre a SD-WAN, Junipe,18 fev. 2023. Disponivel em: https://www.juniper.net/br/pt/research-topics/sd-wan-explained.html. Acesso em: 29 nov. 2023

Oque é SD-WAN, ArubaNetworks, 3 set. 2023. Disponivel em: https://www.arubanetworks.com/br/faq/o-que-e-sd-wan/. Acesso em: 29 nov. 2023

COMER, Douglas E. Redes de Computadores e Internet. 6ª ed. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=1nwdDAAAQBAJ&oi=fnd&pg=PR1&dq=rede+de+computadores&ots=DteNJ1osy8&sig=U9F1czq_BzcneXk81bsinXqSg4#v=onepage&q&f=false. Acesso em: 29 nov. 2023

Oque é o Protocolo de internet, CloudFlare, 14 ago. 2023. Disponivel em: https://www.cloudflare.com/pt-br/learning/network-layer/internet-protocol/. Acesso em: 29 nov. 2023.

DURBANO, Vinicius. SD-WAN: conceito, aplicações e tudo que você precisa saber sobre. 9, mai. 2023. Disponivel em: https://blog.ecoit.com.br/sd-wan/. Acesso em: 29 nov. 2023